

Handbook of Bayesian Deep Learning

BayesAI Consortium

Contents

I	Sampling methods	1
1	Introduction to sampling	2
1.1	Introduction	2
1.2	Monte Carlo	2
1.3	Rejection Sampling	4
1.4	Importance Sampling	4
1.5	Markov chain Monte Carlo	5
1.5.1	Metropolis–Hastings	6
1.5.2	Hamiltonian Monte Carlo	9
1.6	Sequential Monte Carlo	12
1.7	Summary	14
2	Stochastic Gradient MCMC	16
2.1	Introduction	16
2.2	Classic SG-MCMC Algorithms	16
2.2.1	Notation	16
2.2.2	Stochastic Gradient Langevin Dynamics	17
2.2.3	Stochastic Gradient Hamiltonian Monte Carlo	17
2.3	Practical Concerns and Recent Improvements	18
2.3.1	Sampling from Diverse Local Minima	18
2.3.2	Sampling from Wide and Robust Local Minima	18
3	Simulation-based inference	21
3.1	Introduction	21
3.2	Simulation-based inference with Neural Posterior Estimation	22
3.2.1	Neural Posterior Estimation	22
3.2.2	Conditional density estimation with normalizing flows	24
3.2.3	Learning from high-dimensional data with embedding networks	26
3.3	Amortized and sequential inference	27
3.4	Evaluating the correctness of SBI methods	27
3.4.1	The problem of evaluating SBI methods	27
3.4.2	Coverage diagnostics	28
3.4.3	Other methods	28
3.5	Alternative methods	29
3.5.1	Likelihood estimation	29
3.5.2	Likelihood-to-evidence ratio estimation	29
3.5.3	Sampling from the approximate posterior using unnormalised inference models	30
3.5.4	Estimating multiple quantities at once	31
3.6	Current developments	31
3.6.1	Better generative models	32
3.6.2	Improving the flexibility of simulation-based inference	32
3.6.3	Improving robustness to model misspecification	32
3.6.4	Conservative and calibrated posterior approximations	32
3.7	Discussion	33
4	Sequential Monte Carlo samplers	34

4.1	Motivation	34
4.2	Introduction	34
4.3	SMC sampler	35
4.4	MLSMC for TNN	36
4.4.1	MLMC method	37
4.4.2	Trace class neural networks	38
4.4.3	MLSMC TNN method	40
4.4.4	Numerical experiments	41
4.5	Parallel SMC	43
4.5.1	Parallel MCMC	43
4.5.2	Theoretical Result	43
4.5.3	Numerical experiments	43
5	Low-precision sampling	46
5.1	Low-Precision Deep Neural Network	46
5.2	Low-Precision SG-MCMC	47
5.2.1	Preliminary	47
5.2.2	Low-Precision Stochastic Gradient Langevin Dynamics	48
5.2.3	Low-Precision Stochastic Gradient Hamiltonian Monte Carlo	53
5.2.4	Advantage of LP Stochastic Gradient HMC under Non-Convexity	54
5.3	Post-Training Low-Precision Sampling	55
5.3.1	Quantized Pre-Trained Bayesian models	55
5.3.2	Low-Bit Sub-Distribution	57
5.4	Conclusions and Discussions	57
6	Discrete sampling	58
6.1	Introduction	58
6.2	Notation	58
6.3	Locally Balanced Proposals	59
6.4	Gradient-Based Discrete Sampling	60
6.4.1	Gibbs with Gradient	60
6.4.2	Discrete Langevin Proposal	62
6.5	Beyond Locally Balanced Proposals	64
6.5.1	Any-scale Balanced Sampler	65
6.5.2	Automatic Cyclical Sampling	66
6.6	Conclusion	67
II	Laplace approximations	69
7	Introduction	70
7.1	Laplace approximations are local and differential	70
7.2	What is to like about Laplace?	71
7.3	Practical Considerations	72
7.3.1	Isn't this too simplistic?	73
7.3.2	Isn't this too expensive?	73
7.4	All approximations are wrong	73
8	Linearised Laplace	75
8.1	Linearisation	75
8.1.1	Advantages of Using the Linearised Network	77
8.2	The Prior	78
8.3	The Fisher	79
8.4	Scalable Curvature Approximations	83
8.4.1	Low-rank GGN Approximations	83
8.4.2	Kronecker-Factored Approximate Curvature	84
8.5	The Predictive	85

9	Differential geometry	88
9.1	The GGN Metric	88
9.2	The Fisher Metric	90
9.3	Geometric Laplace Approximations	92
10	Analytic functionality	93
10.1	Fixing overconfidence with calibrated uncertainty	93
10.1.1	Provable overconfidence in ReLU-like neural networks	93
10.1.2	Bayesian perspective as a counter measure	94
10.1.3	The role of the prior variance	95
10.1.4	How to mitigate the problem even further	96
10.2	Fixing catastrophic forgetting in continual learning	97
10.2.1	A Bayesian solution	97
10.2.2	Bayesian filtering in weight space	97
10.2.3	From EWC to OSLA to Lo-Fi	98
10.3	Data-centric objectives	99
10.3.1	Active learning: selecting what to label next	99
10.3.2	Bayesian optimisation: finding the optimal point	100
11	Model selection with the marginal likelihood	102
11.1	Marginal likelihood for model selection	102
11.1.1	Hierarchical Bayes	102
11.1.2	Occam’s razor	103
11.1.3	Cross-validation	103
11.1.4	PAC-Bayes	104
11.2	Differentiable marginal likelihood with Laplace	104
11.2.1	Computational considerations	105
11.2.2	Post-hoc vs. online model selection	105
11.2.3	Simple example	106
11.2.4	On the risk of overfitting	107
11.2.5	Connection to flat minima	107
11.3	Use cases	107
11.3.1	Feature selection	108
11.3.2	Neural network pruning	108
11.3.3	Invariance learning	109
11.3.4	Probing	109
III	Variational inference	110
12	Introduction to variational inference	111
12.1	A Variational Approach: Bayesian Inference as Optimisation	111
12.1.1	The Kullback–Leibler Divergence	111
12.1.2	Rewriting the KL Objective	112
12.1.3	The Evidence Lower Bound (ELBO)	112
12.1.4	Interpreting the ELBO	112
12.1.5	Other Lower Bounds on the Log-Evidence	112
13	Gaussian mean-field variational inference	114
13.1	Coordinate Ascent Variational Inference	114
13.2	Stochastic Gradient Optimisation	115
13.2.1	The Reparameterisation Gradient Estimator	115
13.3	Examples of Gaussian Mean-Field Variational Inference	115
13.3.1	Example 1: Bayesian Linear Regression	116
13.3.2	Example 2: Bayesian Neural Networks (BNNs)	116
13.3.3	Example 3: Variational Autoencoders	117
13.4	Limitations of Gaussian Mean-Field Variational Inference	117

14 Dropout-based variational inference	118
14.1 Monte Carlo dropout	118
14.1.1 The dropout objective	118
14.1.2 Gaussian Processes: the cornerstone of the approximation	119
14.1.3 Dropout as a Bayesian approximation	120
14.2 Concrete Dropout: dropout rate as a variational parameter	124
14.3 Dropout injection	125
14.3.1 Dropout injection: the regression case	125
14.3.2 Dropout Injection for classification	126
14.4 Concluding remarks	127
15 Natural-gradient-based variational inference	128
15.1 Natural-Gradient Descent	128
15.1.1 Distinction from Natural Gradient Methods for Maximum Likelihood Estimation	130
15.2 Generalized Bayes & The Bayesian Learning Rule	130
15.2.1 Exponential Family Distributions	130
15.2.2 Bayesian Learning Rule	131
15.2.3 Stochastic Gradient Descent (SGD) from the Bayesian Learning Rule (BLR)	131
15.3 Variational Online Newton Methods	132
15.3.1 VOGN	133
15.3.2 shen2024variational	134
15.4 Concluding Remarks	136
16 Particle-Based Variational Inference	137
16.1 Introduction	137
16.2 Variational Objectives Revisited	138
16.2.1 The KL Divergence and the Evidence Lower Bound (ELBO)	138
16.2.2 Infinitesimal Transport of q by a Vector Field	138
16.2.3 The Derivative of the KL Divergence	139
16.2.4 The Direction of Steepest Descent	139
16.3 Stein Variational Gradient Descent (SVGD)	141
16.3.1 Basic Overview	141
16.3.2 Particle-Based Approximation	142
16.3.3 Complexity and Practical Implementation	143
16.4 Particle-Based Variational Inference (ParVI) as Gradient Flow*	143
16.4.1 Basics of Gradient Flows	144
16.4.2 Wasserstein Gradient Flow	145
16.4.3 Stein Gradient Flow	146
16.4.4 Fisher-Rao and Wasserstein-Fisher-Rao Gradient Flow	147
16.5 Second-Order Methods and Riemannian ParVI*	148
16.5.1 Stein Variational Newton (SVN)	148
16.5.2 Riemannian SVGD (RSVGD)	149
16.6 Stochastic and Scalable ParVI	151
16.6.1 Mini-Batch SVGD	151
16.6.2 Stochastic Particle Optimisation Sampling (SPOS)	152
16.6.3 Random-feature kernels and sub-quadratic interactions	153
16.6.4 GPU and distributed implementations	153
16.7 Applications in Bayesian Deep Learning	153
16.7.1 Bayesian Neural Networks	153
16.7.2 Large-Scale Models and Bayesian Fine-Tuning	154
16.8 Theoretical Guarantees and Open Problems*	155
16.8.1 Continuous Time, Infinite-Particle SVGD	155
16.8.2 Discrete-Time, Infinite-Particle SVGD	156
16.8.3 Finite-Particle SVGD	157
16.8.4 Curse-of-Dimensionality	158
16.8.5 Open problems	158
16.9 Summary	159

17 Variational inference in function space	160
17.1 Introduction	160
17.2 Function-Space Bayes Theorem	160
17.2.1 Setting and Notation	161
17.2.2 Function-Space Bayes Theorem	161
17.2.3 Finite-Dimensional (Evaluation) Form	161
17.3 Function-Space KL and Variational Objectives	162
17.3.1 Measure-theoretic KL divergence	162
17.3.2 Function-space ELBO	162
17.3.3 Well-posedness of ELBO maximisation	162
17.4 Supremum characterisation of the ELBO	163
17.4.1 Adversarial functional VI	163
17.4.2 Sampling-based functional VI	164
17.4.3 Grid-functional divergence	164
17.5 Parameter pushforward measures	164
17.5.1 Data-processing inequality	165
17.5.2 Estimating function-space KL for parameter pushforwards	165
17.5.3 Interpretable priors and the singularity issue	166
17.6 Generalised Variational Objectives	166
17.6.1 Motivation	166
17.6.2 Generalised variational objective (GVO)	166
17.6.3 Wasserstein-2 GVO for Gaussian measures on function spaces	166
17.6.4 Generalised Wasserstein inference with a neural mean (GWI-net)	167
17.7 Discussion	168
18 Sequential variational inference	169
18.1 Introduction	169
18.2 State Space Models	169
18.2.1 Modelling assumptions	170
18.2.2 Examples	170
18.2.3 Linear Dynamical Systems	171
18.2.4 Nonlinear dynamical systems	171
18.2.5 SSMs for online Bayesian deep learning	171
18.3 Online Inference	172
18.3.1 Bayesian filtering	172
18.3.2 Kalman filtering	172
18.3.3 Extended Kalman filtering	173
18.3.4 Statistical Linear Regression	174
18.3.5 Variational Inference	175
18.3.6 State estimation vs parameter estimation	176
18.4 Bayesian Online Natural Gradient	176
18.4.1 Background	177
18.4.2 Update step	177
18.4.3 Computing the expectations	178
18.4.4 Computational complexity	179
18.4.5 Diagonal case	180
18.4.6 Low-rank case	180
18.4.7 Connections to second-order optimisation	180
18.5 Conclusions	181
IV Ensemble methods	182
19 Introduction to ensemble methods	183
19.1 Deep ensembling procedure	183
19.2 Deep ensembles as approximate Bayesian inference	184
19.3 Developments of deep ensembles	185

19.4	Are deep ensembles necessary?	186
19.5	Summary	186
20	Hybrid ensemble sampling methods	188
20.1	On the Efficacy of Optimization-Informed Initialization	188
20.2	Parallelized Exploration for Mode Discovery	189
20.3	Variations of Hybrid Sampling	189
20.3.1	MAP Initialization and Tempered Transitions	191
20.3.2	Ensemble-Based Initialization for Multi-Modality	191
20.3.3	Related and Emerging Techniques	192
20.4	Limitations	192
20.4.1	Weighting across Modes	192
20.4.2	Computational and Resource Costs	193
20.4.3	Fidelity of Mode Discovery	193
20.4.4	Challenges of Local Exploration	193
21	Practical applications of deep ensembles	194
21.1	Introduction	194
21.2	Applications to Large Language Models	195
21.2.1	Mixture-of-Experts	195
21.3	Applications to reinforcement learning	197
21.3.1	Reinforcement Learning from Human Feedback	197
21.3.2	Model Ensemble in Model-Based RL	199
21.4	Applications to weather forecasting	199
21.5	Applications to medical data	200
21.6	Applications to autonomous driving	201
21.7	Patents on deep ensembles	202
22	PAC-Bayesian deep neural network ensembles	203
22.1	A basic introduction to PAC-Bayesian analysis	203
22.1.1	Similarities and differences between PAC-Bayesian analysis and Bayesian learning	205
22.1.2	PAC-Bayes-kl inequality	206
22.2	Ensemble classifiers and weighted majority vote	209
22.3	Oracle bounds for the weighted majority vote	209
22.4	Empirical bounds for the weighted majority vote	212
22.5	PAC-Bayesian deep ensembles	213
22.5.1	Construction of an ensemble	213
22.5.2	Optimising ensemble weighting through minimisation of a second-order PAC-Bayesian bound	214
22.5.3	Empirical example	214
22.5.4	Ensemble pruning	215
22.6	Summary	215
V	Kernel methods	217
23	Introduction to kernel methods	218
23.1	Kernel as Feature Map	218
23.1.1	From Input Space to Feature Space	218
23.1.2	Kernels as Inner Products in Feature Space	219
23.1.3	Example: Quadratic Feature Map	219
23.1.4	The Computational Challenge	220
23.2	Kernel Functions	220
23.2.1	Positive semi-definite kernels	221
23.2.2	Kernel as a Similarity Measure	222
23.2.3	Kernel Compositions and Operations	222
23.2.4	Spectral Perspective	222
23.3	The Kernel Trick	223
23.3.1	Feature Maps and Implicit Representations	223

23.3.2	Reproducing Kernel Hilbert Spaces	223
23.3.3	Computational Advantage and the Representer Theorem	224
23.4	Gaussian Processes	224
23.4.1	Posterior Inference	225
23.5	Limitations	225
23.5.1	Fixed Representations and Expressiveness	225
23.5.2	Computational Scalability	226
23.5.3	The Path Forward: Deep and Adaptive Gaussian Processes	226
24	Deep kernel learning	228
24.1	Review of Gaussian Processes	228
24.2	Deep Kernel Learning	229
24.3	Stochastic Variational Deep Kernel Learning	230
24.3.1	Details of SVDKL	230
24.4	Kernel Estimation	231
24.5	DKL in Practice, Implementations, and Resources	232
25	Deep Gaussian processes	233
25.1	Composing functions and processes	233
25.1.1	Function composition for deep learning	233
25.1.2	Weight matrix factorisation	234
25.1.3	Process composition with latent variables	234
25.1.4	Inference challenges and expressiveness	235
25.1.5	Characteristics of deep GPs as priors	237
25.1.6	Modelling bottleneck layers	238
25.1.7	Outline for the rest of this chapter	239
25.2	Deep Gaussian processes with latent variables	239
25.2.1	Probabilistic definition	240
25.2.2	Variational inference	241
25.2.3	Supervised learning with noisy bottleneck layers	243
25.3	Scalable deep GPs	243
25.4	Further inference approaches	244
25.4.1	Expectation propagation	244
25.4.2	Random feature approximations	245
25.4.3	Sampling-based and structured variational inference	245
25.5	Illustration	246
26	Deep kernel processes and deep kernel machines	250
26.1	Introduction	250
26.2	Background	250
26.2.1	Gaussian Processes	250
26.2.2	Variational Inference for Gaussian Process Classification	252
26.2.3	Sparse Variational Gaussian Processes	255
26.2.4	Deep Gaussian Processes	256
26.3	Deep Kernel Processes	257
26.3.1	Constructing a Deep Kernel Process	257
26.3.2	Bayesian Inference in a Deep Wishart Process	258
26.4	Deep Kernel Machines	259
26.4.1	Infinite-Width Neural Network Limits	260
26.4.2	Standard infinite-width limit	260
26.4.3	The Bayesian representation learning limit and DKMs	261
26.4.4	Sparse DKMs	262
27	Meta-learning deep kernel Gaussian processes	267
27.1	Introduction	267
27.2	Preliminaries	268
27.2.1	Meta-learning and few-shot learning	268
27.2.2	Deep kernel Gaussian processes	268

27.3	Adaptive Deep Kernel Fitting with Implicit Function Theorem	269
27.3.1	The general ADKF-IFT framework with bilevel optimisation	269
27.3.2	Efficient meta-training algorithm for ADKF-IFT	269
27.3.3	ADKF-IFT as a unification framework	270
27.3.4	Highlighted instantiation of ADKF-IFT	271
27.4	Connection to other methods	272
27.4.1	Deep kernel Gaussian processes	272
27.4.2	Meta-learning	273
27.4.3	Multi-Task Gaussian processes	273
27.4.4	Implicit function theorem in machine learning	273
27.5	Empirical evaluation	273
27.5.1	Few-shot molecular property prediction on FS-Mol	273
27.5.2	Out-of-domain molecular property prediction and optimisation	274
27.6	Summary	276

VI Priors 278

28 Introduction to priors 279

28.1	Introductory overview of priors for BDL	280
28.1.1	Informative Priors	280
28.1.2	Uninformative, Reference, and Objective Priors	281
28.1.3	ℓ_2 Regularisation Priors	282
28.1.4	Sparsity-Inducing and Variable Selection Priors	285
28.1.5	Model Priors and Bayesian Model Selection and Averaging	287
28.2	Summary and Outlook	287

29 Statistical properties of priors 288

29.1	Introduction	288
29.1.1	Simple statistical properties of priors	288
29.1.2	Statistical properties of priors in BNNs	289
29.1.3	Width and depth limits of BNNs and their implications	289
29.1.4	Outline	290
29.2	Single-hidden-layer networks	290
29.2.1	First-hidden-layer outputs	290
29.2.2	Combining the outputs of a single-hidden-layer neural network	291
29.2.3	Gaussian process limit for shallow neural networks	291
29.3	Deep neural networks	292
29.3.1	Sub-Weibull characterisation	292
29.3.2	Meijer G -functions characterisation	293
29.3.3	Gaussian process limit for deep neural networks	294
29.4	Different priors	295
29.4.1	α -stable priors	295
29.4.2	Trace-class priors	295
29.5	Different architectures	296
29.5.1	Different activation functions	296
29.5.2	Different layers	296
29.6	Influence of the prior on the posterior	297
29.6.1	Bayesian asymptotics	297
29.6.2	Posterior contraction rates	298
29.6.3	Adaptation	298
29.6.4	Bernstein–von Mises theorems	298
29.7	Discussion and open questions	299

30 Informative priors 300

30.1	A Family of Informative Data-Driven Priors	300
30.2	Uncertainty-Aware Data-Driven Priors	301
30.3	Group-Aware Data-Driven Priors	301

30.4 Summary	302
31 Function-space priors	303
31.1 From priors over parameters to priors over functions	303
31.1.1 Function-space priors are desirable but intractable	304
31.1.2 Interpretation of function-space priors as a change of variables	304
31.1.3 Connections with Neural Processes	305
31.1.4 Sampling from function-space priors	306
31.2 Pathologies of function-space priors	306
31.2.1 Possible priors over θ	306
31.3 Matching neural network priors with target function-space priors	306
31.3.1 Candidates for function-space priors	306
31.3.2 Matching problem and objectives	308
31.3.3 Imposing GP Priors on Bayesian Neural Networks using Wasserstein Distance	309
31.3.4 Examples on a 1D Regression Synthetic Dataset	310
VII Identifiability and symmetries	313
32 Symmetries in Bayesian neural networks	314
32.1 Background: symmetries and invariance	315
32.1.1 Basic definitions	315
32.1.2 Symmetric groups	316
32.1.3 Group actions, invariant functions, and equivariant functions	317
32.2 Going deep into weight-space symmetries in neural networks	318
32.2.1 Weight-space symmetries in neural networks	318
32.2.2 Weight-space symmetries on special layers	320
32.3 From likelihood symmetries to posterior symmetries	326
32.4 Choosing a representative modulo symmetries	327
32.4.1 Scaling symmetry and its representative	328
32.4.2 Permutation symmetry and its representative	328
32.4.3 Combining scaling and permutation representatives	329
32.5 Mode connectivity	329
32.6 Conclusion	332
33 Symmetries and sampling	333
33.1 Posterior Landscape in the Presence of Symmetries	333
33.1.1 Posterior Landscape	333
33.1.2 Impact of Symmetries	334
33.1.3 Prior Influence	335
33.2 Sampling Strategies	336
33.2.1 Symmetry Removal	336
33.2.2 Subspace Inference	336
33.2.3 Multi-Start Sampling	336
33.3 Sampling diagnostics in the presence of symmetries	337
33.3.1 Diagnostics in Parameter Space	337
33.3.2 Diagnostics in Function Space	338
33.3.3 Towards More Meaningful Diagnostics	338
34 Learning inductive bias and symmetries	340
34.1 The Problem Setting	341
34.2 Potential Methods	342
34.2.1 Minimising the Training Loss: Flexible Models Fit Any Data, But Generalise Differently	342
34.2.2 Crossvalidation: The Common Solution	343
34.2.3 When Training Losses <i>do</i> Identify Inductive Bias	344
34.2.4 Explicit Regularisation of Hyperparameters	345
34.2.5 Differentiable Validation Losses and Neural Architecture Search	346
34.2.6 The Need for Something Better	346

34.3	From Bayesian Hierarchical Inference to Model Selection	347
34.3.1	Uncertainty over the Weights: The Typical Bayesian Deep Learning Case	347
34.3.2	Inferring Hyperparameters through Hierarchical Bayes	347
34.3.3	Maximum Marginal Likelihood as a Practical Approximation	348
34.3.4	Practical Approximations	349
34.3.5	Applying Bayesian Model Selection	350
34.4	Why & When Bayesian Model Selection Works	350
34.4.1	Losses During On-Line Prediction	351
34.4.2	Occam's Razor	352
34.4.3	Relationship to Other Frameworks: Crossvalidation, Generalisation Bounds, Compression	352
34.5	Problems with Bayesian Model Selection	353
34.5.1	Sensitivity to Weight Prior	353
34.5.2	Overfitting when Fitting Many Hyperparameters	354
34.5.3	Architectures are Defined	354
34.6	Exact symmetry learning in basis function model	355
34.6.1	Example: Flip symmetry.	355
34.7	Learning invariance in a shallow network	357
34.7.1	Transformation-averaged function construction	357
34.7.2	Example: affine transformations of 2D images	357
34.7.3	Variational objective for a shallow network	358
34.7.4	Experiments: learning useful invariances	359
34.8	Learning invariances in deep neural networks	361
34.8.1	Differentiable marginal likelihood via the Laplace approximation	361
34.8.2	Scalable curvature approximations	362
34.8.3	Practical differentiable Laplace objective	362
34.8.4	Experiments: learned input invariance	363
34.8.5	Empirical comparison with explicit regularisation	364
34.9	Learning layer-wise equivariances in deep neural networks	364
34.9.1	Exact equivariance and group convolution	366
34.9.2	Relaxation through nonstationary kernels	367
34.9.3	Relaxation through residual pathways	368
34.9.4	Experiments: learned layer-wise equivariance	369
34.10	Conclusion and Outlook	370

35 Interpretable inferences via Bayesian pyramids 372

35.1	Introduction	372
35.2	Bayesian Pyramids	372
35.2.1	Model Structure and Identifiability Results	373
35.2.2	Bayesian Inference for Two-layer Bayesian Pyramids	373
35.2.3	Simulation Results and Real Data Application	375
35.3	Bayesian Latent Class Regression with Interpretable Binary Profiles	377
35.3.1	Model Structure and Identifiability Results	378
35.3.2	Bayesian Inference	378
35.3.3	Limiting Behaviour in High-dimensional Clustering	379
35.3.4	Application to Joint Species Distribution Modelling	379
35.4	Bayesian deep generative models for multiplex networks	380
35.4.1	Model Structure and Identifiability Results	381
35.4.2	Bayesian Inference	382
35.4.3	Applications to Brain Connectivity Networks	382
35.5	Going beyond Categorical Data to Arbitrary Data Types: Deep Discrete Encoders	383
35.5.1	Model Setup of DDEs	384
35.5.2	Identifiability and Estimation Consistency of DDEs	385
35.5.3	Scalable Computation Pipeline	386
35.5.4	Simulation Results	387
35.5.5	Diverse Real-world Applications: Images, Texts, Multimodal Data	388
35.6	Concluding Remarks	391

36 Bayesian mixture of experts in BDL	392
36.1 Introduction	392
36.2 Softmax-gated mixture of experts models	393
36.2.1 Regression	393
36.2.2 Classification	394
36.3 Fitting MoE models	395
36.3.1 Markov Chain Monte Carlo sampling	395
36.3.2 Variational inference	396
36.4 Asymptotic properties	396
36.4.1 Density estimation	397
36.4.2 Parameter estimation	399
36.5 Model selection	401
36.5.1 K as a learnable parameter	401
36.5.2 Selection via the ELBO	403
36.6 Extensions and future directions	403
 VIII Scalability	 405
37 Scalable BDL by single forward pass inference	406
37.1 Why do we need scalability in Bayesian Deep Learning?	406
37.2 Bayesian Inference by a single forward pass	408
37.2.1 Formalism	408
37.3 The challenge of a good backbone	409
37.4 The Antidote for the degeneracy: bi-Lipschitzness	410
37.4.1 Bi-Lipschitzness via Two-Sided Gradient Penalty	411
37.4.2 Bi-Lipschitzness via Spectral Normalisation and Residual Connections	412
37.5 The predictive head	412
37.5.1 The standard neural network head	413
37.5.2 The Radial Basis Function (RBF) head	414
37.5.3 The probabilistic head via generative classifiers	414
37.5.4 The Bayesian head via Gaussian Processes	415
37.6 Outlook	417
 38 Scalable Gaussian processes	 419
38.1 Direct approximations for $\mathbf{K}_{\mathbf{xx}}$	419
38.2 Sparse variational Gaussian processes	420
38.3 Decoupled sparse variational Gaussian processes	422
38.4 Interdomain Gaussian processes	423
38.5 Online sparse Gaussian processes	424
38.6 Summary	425
 39 Last-layer inference	 427
39.1 Motivation	427
39.2 Last-layer inference as a Bayesian linear model	428
39.2.1 Likelihood	428
39.2.2 Prior distributions	429
39.2.3 Posterior distributions	431
39.3 Posterior estimation and prediction	432
39.3.1 Markov chain Monte Carlo	432
39.3.2 Variational inference	433
39.3.3 Estimation and prediction	434
39.3.4 Posterior predictive distribution	435
39.3.5 Posterior predictive moments	435
39.4 Application	436
39.4.1 Datasets	436
39.4.2 Implementation details	436
39.4.3 Results	438

39.5	Extensions and limitations	440
39.5.1	Beyond canonical Bayesian linear regression models	440
39.5.2	Further applications	442
39.5.3	Drawbacks of last-layer inference	442
39.6	Final remarks	442
40	Scalable Laplace approximations	443
40.1	Structured Approximations to the Curvature	443
40.1.1	Diagonal approximations	444
40.1.2	Kronecker Factorisations	444
40.1.3	Inference over a subset of parameters	445
40.1.4	Inference over a subset of datapoints	446
40.2	Sampling via Iterative Solvers	447
40.2.1	Sampling by Optimising a Perturbed Objective	447
40.2.2	Iterative Solvers for Sampling	448
40.2.3	Monte Carlo Predictive Distributions	448
40.3	Model Selection in Scalable Laplace Approximations	448
40.3.1	The Marginal Likelihood Optimisation Framework	448
40.3.2	Optimisation via the Evidence Lower Bound (ELBO)	449
40.3.3	Optimisation via the Expectation-Maximisation (EM) Algorithm	449
40.3.4	Practical Hyperparameter Tuning for Scalable Methods	450
40.4	Concluding Remarks	451
41	Learned Compression with Deterministic Codes	452
41.1	What is Compression?	453
41.2	Lossless compression	453
41.2.1	Optimal Symbol Code	454
41.2.2	Lossless Coding and Statistical Estimation	455
41.2.3	Huffman Coding	455
41.2.4	Arithmetic Coding	456
41.2.5	Neural Lossless Compression	457
41.3	Lossy Compression	459
41.3.1	From Vector Quantization to Transform Coding	460
41.3.2	Neural Lossy Compression	461
41.3.3	Connection with latent variable models	463
42	Learned Compression with Stochastic Codes	466
42.1	Lossless Compression with Randomization: Bits-back Coding	466
42.1.1	Invertible Sampling and Asymmetric Numeral Systems	467
42.1.2	A very brief introduction to symmetries	467
42.1.3	Bits-back coding with discrete symmetries: Shuffle Coding	469
42.1.4	Bits-back coding with side-information	470
42.2	Lossy Compression with Randomization: Relative entropy coding	472
42.2.1	The most special case: dithered quantisation	472
42.2.2	General Relative Entropy Coding and Its Limitations	473
42.3	Applications of Relative Entropy Coding	474
42.3.1	Compression with Bayesian implicit neural representations	475
IX	Applications	477
43	Bayesian LLMs	478
43.1	Bayesian fine-tuning of LLMs	478
43.1.1	Evaluation	479
43.1.2	LoRA ensemble	479
43.1.3	Laplace-LoRA	480
43.2	Bayesian LLM Training with Variational Learning	480
43.2.1	Variational Learning for Language Models	480

43.2.2	IVON for LLMs	481
43.2.3	LoRA Fine-tuning with IVON	483
43.3	Language Generation with Predictive Posteriors	483
43.3.1	Uncertainty-Aware Minimum Bayes Risk Decoding with Posteriors	484
43.3.2	Selective Prediction using Expected Risk	485
43.4	Uncertainty Estimation for Autoregressive LLMs	485
43.5	Bayesian Model Merging for LLMs	486
43.5.1	Variational Model Merging	487
43.5.2	Recovering Averaging using Variational Merging	487
43.5.3	Uncertainty-Based Gradient Matching & Data Removal	488
43.6	Concluding Remarks	488
44	Confidence-based selective classification	489
44.1	Background	490
44.2	Mathematical Framework	492
44.2.1	Problem Setup	492
44.2.2	Confidence Scoring for Selective Classification	492
44.2.3	Selective Classification by Thresholding	496
44.2.4	Coverage and Rejection Rate	496
44.2.5	Selective Accuracy and Risk	496
44.2.6	Target Accuracy Thresholding	497
44.2.7	Risk-Coverage Curves	497
44.3	Case Example on CIFAR-100	497
44.4	Conclusion	499
45	Biomedical data imputation	505
45.1	Introduction	505
45.2	Missing data	505
45.2.1	Missing data types	505
45.2.2	Missing data methods	507
45.3	Variational autoencoder	507
45.3.1	Deep latent variable models	507
45.3.2	Variational autoencoders	508
45.3.3	Beta-VAEs	511
45.4	Variational autoencoders and missing data	512
45.4.1	Multiple imputation with beta-VAEs	514
45.4.2	Tuning beta	514
45.5	Biomedical application	515
45.5.1	Dataset and missingness scenarios	515
45.5.2	Experiment design	515
45.5.3	Experimental results	516
45.6	Discussion	517
46	Deep Bayesian spatio-temporal modelling of wildfires	519
46.1	Introduction	519
46.2	Application: Analysis of Extremes in Fire Weather Index with an Extremes-Aware Variational Autoencoder	520
46.2.1	Flexible Max-ID Extremes Model	520
46.2.2	Extremes-Aware Variational Autoencoder (xVAE)	521
46.2.3	Application to the Fire Weather Index	522
46.2.4	Tail-Dependence Validation with Extremal Dependence Measures	524
46.2.5	Posterior Inference: Areal Exceedance Probabilities	524
46.3	Application: Level-Set Dynamics with Bayesian Deep Echo State Network (ESN) for Modelling Fire Front Propagation	526
46.3.1	Level Set Background	526
46.3.2	ESN Background	527
46.3.3	Bayesian Hybrid Level Set/Deep ESN Model	528

46.3.4	Application to the 2018 California Camp Fire	530
46.4	Conclusion	531
47	Software for BDL	533
47.1	Introduction	533
47.2	Probabilistic programming ecosystems used for BDL	534
47.3	Toolkits for approximate Bayesian deep learning	536
47.4	Applications and downstream toolchains	537
47.5	Final remarks	539
X	Topical developments	541
48	Subspace inference	542
48.1	Method overview	542
48.2	Subnetwork inference	542
48.2.1	Expressivity of subnetwork-derived predictive distributions	543
48.2.2	Subnetwork-based Laplace approximation	543
48.3	Subspaces of connected modes	543
48.3.1	Constructing a subspace	544
48.3.2	Subspace posterior inference	545
48.4	Related methods	545
49	Implicitly Bayesian prediction rules	547
49.1	Explicitly vs. Implicitly Bayesian Methods	548
49.1.1	The Bayesian Inferential Approach	549
49.1.2	Implicitly Bayesian Prediction Rules	549
49.2	Characterising Implicit Bayes	550
49.2.1	Conditionally Identically Distributed	551
49.2.2	Stationarity	551
49.2.3	Spreadability	551
49.3	Why and how to make predictors implicitly Bayesian?	552
49.3.1	Turning a prediction rule Bayesian	552
49.3.2	Exchangeable-ified prediction rules are always better	552
49.3.3	Non-implicitly Bayesian agents are vulnerable to adversarial bets	554
49.4	Measuring Implicit Bayesianness	557
49.5	Conclusion	557
50	Diffusion models	559
50.1	Introduction	559
50.1.1	Defining Diffusion Models	559
50.1.2	Context within Probabilistic Generative Models	559
50.1.3	Chapter Objectives and Roadmap	560
50.2	The Forward Process: The Perturbation Kernel	561
50.2.1	A Fixed Markovian Process	561
50.2.2	The Marginal Distribution	561
50.3	The Reverse Process: The Generative Kernel	562
50.3.1	A Parameterised Markovian Process	562
50.3.2	True Reverse Posterior	562
50.3.3	Parameterising the Generative Transition	563
50.4	Optimisation Objective: From an Evidence Lower Bound to a Simplified Denoising Loss	563
50.4.1	The Variational Lower Bound (ELBO)	563
50.4.2	Simplification via Reparameterisation	564
50.4.3	The Simplified Loss Function	565
50.4.4	Training and Sampling Algorithms	565
50.5	Continuous-Time Diffusion: Score-Based Generative Models	566
50.5.1	Convergence to a Stochastic Differential Equation (SDE)	566
50.5.2	The Reverse-Time SDE	568

50.5.3	The Score Function: Unification of Objectives	568
50.5.4	Sampling via SDE and ODE Solvers	569
50.6	Improving Sampling Efficiency and Controllability	570
50.6.1	Accelerated Sampling: Denoising Diffusion Implicit Models (DDIM)	570
50.6.2	Conditional Generation	571
50.6.3	Classifier-Free Guidance	571
50.6.4	Latent Diffusion Models	572
50.7	Diffusion Models for Discrete Data Domains	572
50.7.1	Beyond Gaussian Perturbations	572
50.7.2	Discrete-State Markov Processes	572
50.8	Chapter Summary	576
50.9	Bibliographical Notes and Further Reading	576
51	Prior-data Fitted Networks	578
51.1	From Inference to Prediction	578
51.2	Prior Fitting	579
51.3	Dataset Priors	580
51.3.1	A Bayesian Linear Regression Prior	580
51.3.2	A Bayesian Neural Network Prior with Architectural Uncertainty	581
51.3.3	SCM Extension	582
51.3.4	A Simple Gaussian Process Prior	582
51.3.5	A Fully Bayesian Gaussian Process Prior	583
51.4	PFN Architectures	583
51.4.1	Original PFN Architecture	584
51.4.2	TabPFNv2 Architecture with 2-D Attention	585
51.5	Applications of Prior-data Fitted Networks	586
51.5.1	Tabular Classification and Regression	586
51.5.2	Bayesian Optimisation	586
51.5.3	PFNs for SBI	587
51.5.4	Time Series Forecasting	587
51.5.5	Graphs	587
51.5.6	Causal Interventions	587
51.5.7	Learning Curve Extrapolation	587
51.5.8	Invariant Learning	587
51.5.9	Aleatoric and Epistemic Uncertainty Decomposition	587
52	Advances in singular learning theory for BDL	588
52.1	Setup	589
52.2	Deep learning is singular	591
52.2.1	Do singularities really matter?	592
52.2.2	The learning coefficient	592
52.2.3	Resolution of singularities and mixture of standard forms	594
52.3	Misconception 1: Laplace approximation is valid for singular models	595
52.3.1	Laplace for prediction	595
52.3.2	Laplace for model selection	596
52.4	Misconception 2: Effective dimensionality is captured by the eigenspectrum of the Hessian	597
52.4.1	Local learning coefficient	598
52.5	Misconception 3: The benefits of marginalisation are not about predictive performance	599
52.6	Misconception 4: Good ELBO means good predictive performance	601
52.7	Summary	602
53	Active Learning and Experimental Design	603
53.1	Decision-theoretic foundations	604
53.1.1	Bayesian decision theory	604
53.1.2	Frequentist decision theory	606
53.2	Theoretically optimal experimental design	608
53.2.1	Data-gathering policies	608

53.2.2	Bayesian experimental design	609
53.2.3	Frequentist experimental design	611
53.3	Practical experimental design	612
53.3.1	Expected-uncertainty-reduction approaches	612
53.3.2	Fisher-information and estimator-sharpness approaches	614
53.3.3	Adaptive design	615
53.3.4	Estimation of BED objectives	617
53.4	Active learning	618
53.4.1	Overview	618
53.4.2	Data-gathering methods	620
53.4.3	Methodological considerations	624
53.5	Conclusion	625
54	Causal-aware BDL	626
54.1	Introduction	626
54.2	Basics for causal machine learning	627
54.2.1	Basics and notations	627
54.2.2	Structural Equation Model	628
54.2.3	Structural identifiability	629
54.2.4	Causal inference through SEMs	629
54.2.5	DAG constraints for causal discovery	630
54.3	Bayesian Deep Learning for Causality with Static Data	631
54.3.1	Deep End-to-End Causal Inference	631
54.3.2	BayesDAG: Fully Bayesian treatment through sampling	634
54.4	Bayesian Deep Learning for Causality with Temporal Data	637
54.4.1	Rhino: Deep Causal Temporal Relationship Learning with History-Dependent Noise	638
54.4.2	Continuous-time extension of Rhino: SCOTCH	641
54.5	Further advances in causal modelling	641
55	Credal BDL and Imprecise Probability in BDL	643
55.1	Introduction	643
55.2	Background and Preliminaries	645
55.2.1	Posterior Predictive Distributions and EU	645
55.2.2	Imprecise Probabilities	647
55.2.3	Quantifying and Disentangling Aleatoric and Epistemic Uncertainties	649
55.3	CBDL and Its Properties	650
55.3.1	CBDL algorithm	650
55.3.2	Theoretical Properties of CBDL	653
55.4	Experiments	654
55.4.1	(Predictive) Uncertainty Quantification	655
55.5	Conclusion and Open Problems	658
56	Bayesian Reinforcement Learning	659
56.0.1	Why Traditional Exploration Often Fails	659
56.0.2	Historical Development: From Bandits to Full Environments	659
56.0.3	The Bayesian Framework: Beliefs and Updates	660
56.0.4	The Mathematical Structure of Sequential Decision Making	660
56.0.5	Illustrative Example: Hazardous Navigation	660
56.0.6	Information as a Resource	661
56.1	Exact Bayesian methods: the tabular foundation	661
56.1.1	Posterior Sampling for Reinforcement Learning (PSRL)	661
56.1.2	Bayes-Adaptive Monte Carlo Planner	663
56.1.3	Bayesian Exploration Bonuses (BEB)	665
56.1.4	Synthesis and Practical Guidance	666
56.2	Gaussian Process Methods: Nonparametric Bayesian RL for Continuous Spaces	666
56.2.1	From parametric to Nonparametric Function Approximation	666
56.2.2	GP-SARSA: Value Function Learning with Gaussian Processes	667

56.2.3	PILCO: Probabilistic Inference for Learning Control	667
56.2.4	Connection to Uncertainty Types and Exploration	668
56.3	Neural Network Approximations: Scaling Bayesian RL to High-Dimensional Problems	669
56.3.1	From Exact Inference to Neural Approximation	669
56.3.2	Variational Inference for Bayesian Neural Networks	669
56.3.3	Variational Bayesian Deep Q-Networks	669
56.3.4	Bayesian Successor Features for Exploration	671
56.4	Bayesian RL Methods Comparison	672
56.5	Applications and case studies	672
56.6	Future directions	673

Symbol Description

x	$\in \mathbb{R}^D$, an input to the neural network	$\mathcal{L}_\Omega(\theta)$	$= \mathcal{L}(\theta) + \Omega(\theta)$, regularized empirical risk, energy function
y	$\in \mathbb{R}^C$, a single training target (e.g. a one-hot encoded class label)	$J_\theta f(x, \theta_*)$	$= \left[\frac{\partial f_c(x, \theta_*)}{\partial \theta_p} \right]_{c=1, \dots, C, p=1, \dots, P} \in \mathbb{R}^{C \times P}$, Jacobian of the network with respect to parameters. Pullback by f .
θ	$\in \mathbb{R}^P$, parameters (“weights and biases”) of the neural network	Z	$= \int \exp(-\mathcal{L}_\Omega(\theta)) d\theta \in \mathbb{R}$, the partition function
Θ	The set of all the possible parameters of the neural network	$\mathcal{N}(x; \mu, \Sigma)$	The Gaussian pdf over x , with mean μ , covariance Σ
$\mathcal{D}$	$= \{(x_n, y_n)\}_{n=1, \dots, N}$, training data for supervised problems	$\mathcal{GP}(f; m, k)$	A Gaussian process measure with mean function m and covariance function (kernel) k
X	$\in \mathbb{R}^{D \times N}$, training inputs	$A^\top$	The transpose of the linear operator A . (For arrays, by convention, this reverses the order of axes)
Y	$\in \mathbb{R}^{C \times N}$, training targets	$\mathbb{1}(\cdot)$	Indicator function, which evaluates to 1 if its argument is true and to 0 otherwise
$f(x, \theta)$	$\mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$, “the deep neural network”, a differentiable program		
$\ell(y, y')$	$\mathbb{R}^C \times \mathbb{R}^C \rightarrow \mathbb{R}$, training loss, negative log likelihood for one datum		
Ω	$\mathbb{R}^P \rightarrow \mathbb{R}_+$, regularizer, negative log prior		
$\hat{\mathcal{L}}(\theta, \mathcal{D})$	$= \sum_{n=1}^N \ell(f(x_n, \theta), y_n)$, empirical risk, negative log batch likelihood		

Preface

Statistical analysis and machine learning have often differed in emphasis: the former has traditionally sought to understand data, while the latter has placed greater emphasis on prediction. Understanding data is a worthwhile goal in its own right. However, predictions derive their value from the decisions they inform—a point that is easily lost on the machine learning community, with its focus on benchmarks measuring predictive performance. Indeed, a weather forecast of rain tomorrow, for example, is useful only insofar as it helps us decide whether to carry an umbrella or cancel a picnic.

Once we put decisions front and centre, it becomes natural to ask whether they are rational and, ideally, optimal. Bayesian decision theory provides a principled answer to this question. From this perspective, Bayesian deep learning (BDL) is not primarily about approximating posterior distributions over model parameters, or even posterior predictive distributions. These are means to a more important end: approximating Bayes-optimal decisions—the decisions that an ideal Bayesian agent would make. Deep neural networks are among the most powerful and flexible models we have discovered so far, and using them to approximate such decisions therefore seems to be one of our best available avenues.

One might argue that a sufficiently capable language model, pretrained on the entire internet, will eventually learn to make Bayes-optimal decisions implicitly. Perhaps it will, but current systems do not yet give us sufficient reason to rely on this alone. If we want control over these approximations, and insight into their quality, the most direct route is to perform Bayesian inference within neural network models and draw on the advances made by Bayesian statistics over the past two centuries. This is the basis for BDL, and what this book is about.

While parts of the book develop a principal route through the landscape of Bayesian methods, other chapters cover foundational and mathematical aspects. Scalability asks how Bayesian reasoning can survive the size of contemporary models and datasets. The applications of the book demonstrate how widely these points reach, and topical developments showcase possible future routes BDL approaches may take. This book is consequently both a technical reference and an invitation to view deep learning through a wider lens. Our hope is that readers come away seeing BDL not as a specialised corner of machine learning, but as a framework for building powerful models while remaining explicit about evidence, uncertainty, and the decisions that give predictions their purpose.

Before presenting the content of the book, the topics it covers, and the people who contributed to it, we would like to emphasise that this book is not intended as a comprehensive survey of the field. Many important subjects receive only limited attention or are not covered at all, such as predictive Bayes or connections between Bayesian inference and optimisation duality. Likewise, many researchers whose work has shaped Bayesian deep learning are not represented among the authors. The book should therefore rather be understood as a snapshot of a broad and rapidly developing field at a particular moment in time. Its selection of topics and contributors reflects the scope and circumstances of this collaborative project, and is therefore necessarily biased. Hence, it should not be interpreted as a judgement about the relative importance of these topics. We still hope, nonetheless, that the book will serve as a useful reference for researchers, educators, and practitioners, as well as provide a foundation for future work in Bayesian deep learning.

Reader's guide

How to read this book

You do not have to read this book from front to back. It is written to be used like a reference work—or a wiki—in which almost every chapter is a self-contained entry that you can open on its own. Find the topic you need, read that chapter, and follow its cross-references outwards only as far as your question requires. This guide tells you where to start, what each chapter assumes, and which routes through the book are worth taking.

What this book covers

Bayesian deep learning (BDL) asks how the machinery of Bayesian inference—priors, posteriors, predictive distributions, marginal likelihoods, and the decisions they license—can be brought to bear on models with millions or billions of parameters, for which none of the classical guarantees or algorithms apply unmodified. This book covers that question in ten parts. Five of them develop the major families of approximate inference used in practice: Monte Carlo sampling (Part I), Laplace approximations (Part II), variational inference (Part III), ensembles (Part IV), and kernel and Gaussian-process methods (Part V). Two cover the modelling questions that are assumed as prerequisites by the aforementioned algorithms: what a prior over a neural network actually means (Part VI), and what the symmetries and non-identifiability of neural parameterisations do to inference and interpretation (Part VII). One is devoted to making all of this run at modern scale (Part VIII), one to worked applications and software (Part IX), and the last to topics at the current research frontier, from diffusion models and singular learning theory to causal, credal, and reinforcement-learning extensions (Part X). The parts are largely parallel rather than sequential: they are alternative and complementary answers to the same problem, and comparing them is a large part of what the book is for.

How the book is organised

Several deliberate choices make non-linear reading possible, and it is worth knowing about them before you start.

Parts have entry chapters. Most parts open with an introductory chapter that sets up the problem, fixes the notation for that family of methods, and previews the chapters that follow. These entry chapters are short, self-contained, and written for readers arriving cold. If you want to know what a part is about without committing to it, read only its entry chapter.

Notation is shared across the whole book. A single set of symbols is used throughout, defined by shared macros and summarised in the mathematical glossary in the front matter. Whenever a chapter writes θ for network parameters, $f(x, \theta)$ for the network, or $\mathcal{D}$ for the training data, it means the same thing as every other chapter. If an unfamiliar symbol appears, the glossary is the first place to look.

Chapters are cross-referenced, not chained. Where a chapter uses a result developed elsewhere, it says so and points to the specific chapter. Treat those pointers as hyperlinks rather than as instructions: follow them when you need the derivation, and ignore them when you are happy to take the result on trust.

Coverage is broad by design. The book is a collaborative work by many authors, and different chapters are pitched at different levels—some are tutorial, some survey a literature, some present a specific recent framework in depth. This is a feature for a reference work. It does mean that reading order should follow your interests rather than the page numbers.

Six chapters that unlock the rest

If you are new to the field, the fastest way in is not to start at Chapter 1 and read on. It is to read the following six entry chapters, which between them establish essentially all of the vocabulary the remaining fifty chapters rely on.

- Chapter 12, on the evidence lower bound and inference as optimisation. This is the single most widely reused chapter in the book.
- Chapter 1, on Monte Carlo, Markov chain Monte Carlo, Hamiltonian Monte Carlo, and convergence diagnostics.
- Chapter 7, on the Laplace approximation and why a Gaussian around a trained network is the cheapest useful posterior.
- Chapter 19, on deep ensembles and their relation to Bayesian model averaging.
- Chapter 23, on kernels, reproducing kernel Hilbert spaces, and Gaussian processes—the function-space view used throughout.
- Chapter 28, on what a prior over a neural network is and what it does.

With these six in hand, any other chapter in the book becomes accessible, and the per-chapter prerequisites in Tables 1 and 2 tell you what else, if anything, a given chapter expects.

The ten parts at a glance

Part I, Sampling methods (Chapters 1–6). Asymptotically exact posterior inference by simulation. Chapter 1 covers Monte Carlo, rejection and importance sampling, Metropolis–Hastings, Gibbs, MALA, HMC, diagnostics, and sequential Monte Carlo; Chapter 2 makes MCMC compatible with mini-batching (SGLD, SG-HMC, cyclical schedules); Chapter 3 treats likelihood-free inference for black-box simulators; Chapter 4 develops multilevel and parallel SMC samplers; Chapter 5 asks what low-precision arithmetic does to a Markov chain; and Chapter 6 covers gradient-based samplers on discrete spaces. Start at Chapter 1; the remaining five chapters can then be read in any order, and Chapter 3 is nearly independent of the rest.

Part II, Laplace approximations (Chapters 7–11). The cheapest route from an already-trained network to a posterior: a Gaussian centred at the optimum with curvature-derived covariance. Chapter 7 motivates and derives it; Chapter 8 turns it into a Gaussian process over functions and introduces the generalised Gauss–Newton matrix and its scalable approximations; Chapter 9 views the same object through the lens of Riemannian and information geometry; Chapter 10 extracts closed-form predictive and decision-theoretic quantities from it; and Chapter 11 uses the resulting marginal likelihood for model selection. Read the first two in order, then whichever of the last three you need.

Part III, Variational inference (Chapters 12–18). Inference recast as optimisation over a family of distributions. Chapter 12 derives the evidence lower bound; Chapter 13 develops the Gaussian mean-field workhorse, coordinate ascent, and the reparameterisation gradient; Chapter 14 shows that dropout at test time is a variational approximation; Chapter 15 exploits the geometry of distribution space and derives practical optimisers such as VON, VOGN, and IVON; Chapter 16 replaces the parametric family with interacting particles (SVGD and gradient flows); Chapter 17 moves the variational problem into function space; and Chapter 18 handles streaming and non-stationary data through filtering. Read Chapters 12–13 in order; the remaining five are independent branches.

Part IV, Ensemble methods (Chapters 19–22). The most widely deployed approach in practice, and the one with the lightest prerequisites. Chapter 19 explains why retraining from different initialisations approximates Bayesian model averaging; Chapter 20 combines ensembling with sampling to explore multimodal posteriors; Chapter 21 surveys applications from mixtures of experts to weather forecasting; and Chapter 22 gives PAC-Bayesian generalisation bounds for weighted majority votes. Readers who want to see something work before studying inference machinery can begin here.

Part V, Kernel methods (Chapters 23–27). The function-space tradition, and the bridge between deep networks and exact Bayesian inference. Chapter 23 covers feature maps, kernels, RKHS, and Gaussian processes; Chapter 24 learns the kernel with a neural network (deep kernel learning); Chapter 25 composes Gaussian processes into deep Gaussian processes; Chapter 26 develops deep kernel processes and machines together with the infinite-width limit; and Chapter 27 meta-learns deep kernels for small-data tasks. Chapter 23 first; the rest branch from it.

Part VI, Priors (Chapters 28–31). What a prior over weights actually says about functions. Chapter 28 surveys the main families and the role of the prior predictive; Chapter 29 studies their statistical properties, including tail behaviour, infinite-width limits, and posterior contraction; Chapter 30 constructs informative, data-driven priors; and Chapter 31 specifies priors directly in function space and matches weight-space priors to them. This part is short and unusually high-leverage: it explains a great many empirical observations reported elsewhere in the book.

Part VII, Identifiability and symmetries (Chapters 32–36). Neural parameterisations are massively redundant, and this has consequences. Chapter 32 catalogues weight-space symmetries and mode connectivity; Chapter 33 works out what they do to samplers and to convergence diagnostics; Chapter 34 turns the question around and learns invariances and inductive biases from data; Chapter 35 constructs deep generative models that are identifiable by design; and Chapter 36 analyses Bayesian mixtures of experts with softmax gating. Start at Chapter 32; the last three chapters are independent of one another.

Part VIII, Scalability (Chapters 37–42). Making the preceding methods affordable. Chapter 37 covers methods that produce uncertainty from one forward pass; Chapter 38 covers sparse, inducing-point, and spectral approximations for Gaussian processes; Chapter 39 restricts inference to the final layer; Chapter 40 scales the Laplace approximation through structured curvature, function-space duality, and iterative solvers; and Chapters 41 and 42 form a self-contained pair on Bayesian data compression, with and without quantisation. This is the part practitioners tend to need soonest.

Part IX, Applications (Chapters 43–47). Worked end-to-end problems. Chapter 43 covers Bayesian treatment of large language models, including Bayesian fine-tuning and model merging; Chapter 44 evaluates uncertainty quantification methods head-to-head through selective classification; Chapter 45 treats missing biomedical data with deep generative models; Chapter 46 develops hybrid Bayesian hierarchical and deep models for wild-fire extremes and spread; and Chapter 47 surveys the software ecosystem. Any of these can be read early, as motivation. If you intend to implement something, Chapter 47 is worth reading first.

Part X, Topical developments (Chapters 48–56). Nine largely independent chapters at the research frontier: low-dimensional subspace inference (Chapter 48); a behavioural characterisation of what makes a predictor Bayesian (Chapter 49); diffusion models (Chapter 50); prior-data fitted networks and amortised in-context prediction (Chapter 51); singular learning theory and what it corrects about the standard asymptotic picture (Chapter 52); active learning and Bayesian experimental design (Chapter 53); causal-aware BDL (Chapter 54); credal sets and imprecise probability for separating aleatoric from epistemic uncertainty (Chapter 55); and Bayesian reinforcement learning (Chapter 56). Read whichever interests you; none of them requires another.

A map of the parts

Figure 1 shows how the parts depend on one another. It is a map, not a timetable: an arrow records that the target part will make more sense if you have seen the source, not that you are forbidden from starting elsewhere. Three parts have no incoming arrows and are therefore legitimate places to begin—sampling, variational inference, and kernel methods—and each corresponds to one of the three distinct ways this book views a posterior: as something

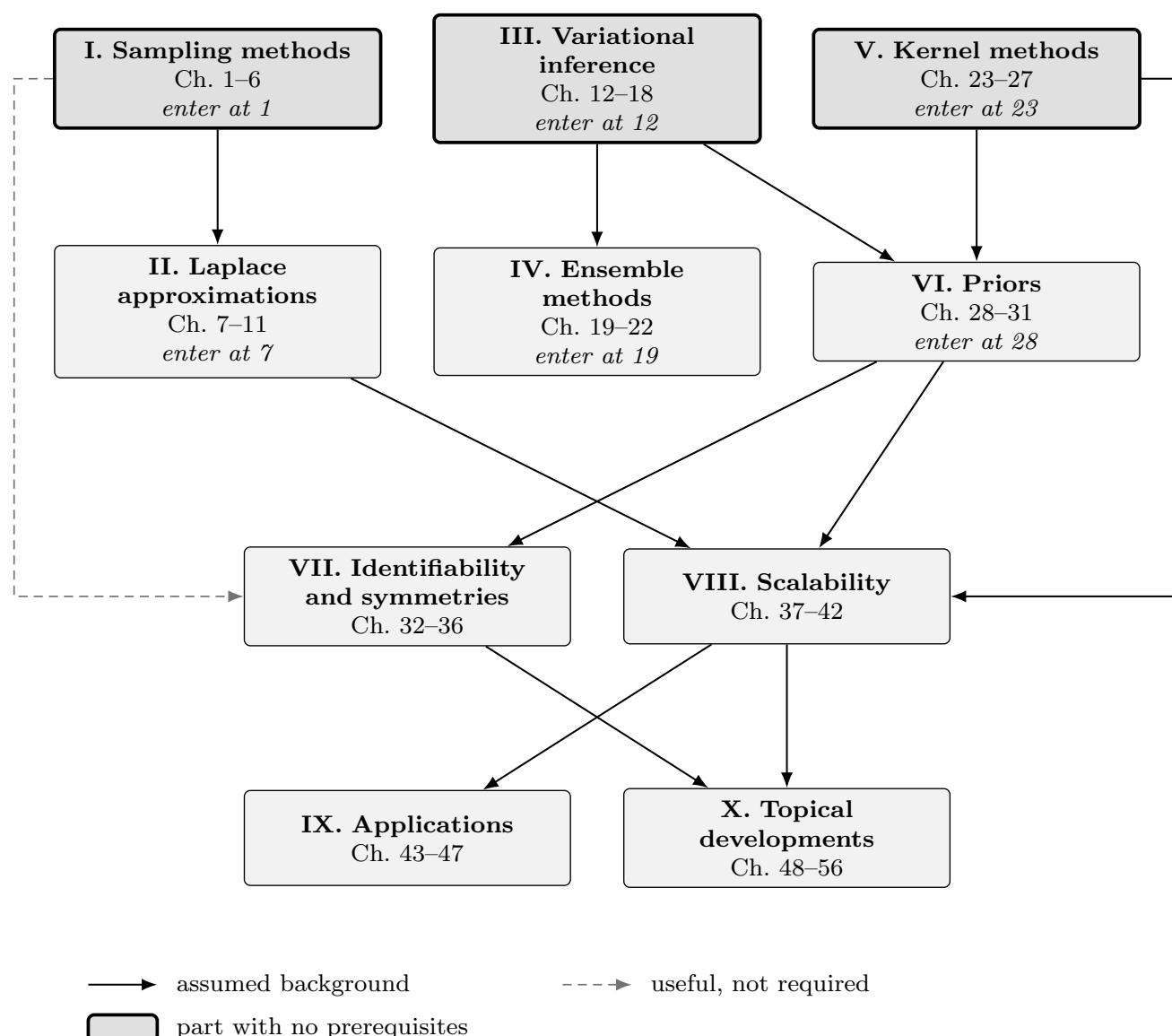


Figure 1: Recommended dependences between the ten parts. Solid arrows mark background the target part assumes; dashed arrows mark material that enriches it without being required. Thick-bordered parts have no prerequisites and can be read first. For legibility only the strongest links are drawn: in particular Part III (variational inference) supplies machinery—the evidence lower bound above all—that is reused to some degree in every part that follows it, and Part IX draws on whichever method its case study happens to use. Chapter-level prerequisites are given in Tables 1 and 2.

to simulate, as something to optimise, and as something to specify directly over functions. Choosing a starting point among them is largely a matter of taste.

Chapter-level prerequisites

Tables 1 and 2 give, for every chapter, the small number of chapters worth reading first and a second list of chapters that deepen it. The first list is deliberately short: it names what a chapter genuinely leans on, not everything related to it. Chapters with an empty first column need nothing beyond a standard background in probability, linear algebra, and deep learning. Used together with the map in Figure 1, these tables let you assemble your own route: pick the chapter you actually want, take the transitive closure of its prerequisites, and read that—usually two or three chapters rather than forty.

Table 1: Chapter-level prerequisites for Parts I–V. “Read first” lists the chapters a chapter leans on; “also useful” lists chapters that deepen it but are not required.

Ch.	Topic	Read first	Also useful
<i>Part I: Sampling methods</i>			
1	Introduction to sampling	—	—
2	Stochastic gradient MCMC	1	5
3	Simulation-based inference	1	12, 50, 51
4	Sequential Monte Carlo samplers	1	31, 19
5	Low-precision sampling	1, 2	41
6	Discrete sampling	1	2
<i>Part II: Laplace approximations</i>			
7	Introduction to Laplace approximations	—	Part I, for contrast
8	Linearised Laplace approximations	7	23, 15
9	Differential-geometric perspective	8	15, 32
10	Analytic predictive functionality	7, 8	53, 44
11	Model selection via marginal likelihood	7, 8	40, 34
<i>Part III: Variational inference</i>			
12	Introduction to variational inference	—	—
13	Gaussian mean-field VI	12	—
14	Dropout-based VI	12	24, 19
15	Natural-gradient VI	12, 13	8, 43
16	Particle-based VI	12, 1	2, 50
17	VI in function space	12, 23	31, 38
18	Sequential VI	12, 15	1, 8
<i>Part IV: Ensemble methods</i>			
19	Introduction to deep ensembles	—	12
20	Hybrid ensemble sampling	19, 2	32, 33
21	Practical applications of ensembles	19	36, 47
22	PAC-Bayesian ensembles	19	12, 49
<i>Part V: Kernel methods</i>			
23	Introduction to kernel methods	—	—
24	Deep kernel learning	23	37, 39
25	Deep Gaussian processes	23	12, 38
26	Deep kernel processes and machines	23, 25	31
27	Meta-learning deep kernel GPs	23, 24	53

Table 2: Chapter-level prerequisites for Parts VI–X. Conventions as in Table 1.

Ch.	Topic	Read first	Also useful
<i>Part VI: Priors</i>			
28	Introduction to priors	—	12, 23
29	Statistical properties of priors	28	25, 52
30	Informative priors	28	13, 39
31	Function-space priors	28, 23	17
<i>Part VII: Identifiability and symmetries</i>			
32	Symmetries in Bayesian neural networks	—	28
33	Symmetries and sampling	32, 1	2, 48
34	Learning inductive bias and symmetries	32, 11	13, 40
35	Identifiable deep generative models	32	52
36	Bayesian mixtures of experts	32	1, 13
<i>Part VIII: Scalability</i>			
37	Single-forward-pass inference	24, 19	38, 39
38	Scalable Gaussian processes	23	12, 15
39	Last-layer inference	38, 28	8, 2
40	Scalable Laplace approximations	7, 8	38, 17
41	Bayesian compression with quantisation	12, 13	5
42	Compression with stochastic codes	41	1, 50
<i>Part IX: Applications</i>			
43	Bayesian large language models	15	8, 39
44	Selective classification	—	8, 19, 53
45	Biomedical data imputation	12, 13	30
46	Spatio-temporal modelling of wildfires	12, 13	19
47	Software for BDL	—	all method parts
<i>Part X: Topical developments</i>			
48	Subspace inference	8, 39	33, 40
49	Implicitly Bayesian prediction rules	12	17, 43
50	Diffusion models	12, 13	2
51	Prior-data fitted networks	3	27, 43
52	Singular learning theory	28, 32	7, 29
53	Active learning and experimental design	12	10, 56
54	Causal-aware BDL	12, 13	2, 50
55	Credal BDL and imprecise probability	19	44, 53
56	Bayesian reinforcement learning	12, 23	39, 53

Suggested routes through the book

The following itineraries are ways of reading the book that we have found coherent. Each is self-contained in the sense that its chapters supply one another's prerequisites.

A first pass through the field (about twelve chapters). Chapters 12, 13, 1, 2, 7, 8, 19, 23, 24, 28, 39, 44. This covers all five families of approximate inference treated in the book, the prior, one scalable method, and one empirical comparison of the lot. It is also a reasonable skeleton for a one-semester graduate course, with Chapter 47 added for practical sessions.

I have a trained network and need calibrated uncertainty. Chapters 7, 8, 40, 39, 19, 37, 44, and 47. This is the shortest path from a point estimate to a usable posterior predictive, and it avoids retraining throughout.

Sampling and Monte Carlo. Chapters 1, 2, 4, 5, 6, 20, 16, and 33. The last two are the payoff: particle methods sit between sampling and optimisation, and the symmetry chapter explains why diagnostics on neural posteriors can mislead.

Variational and optimisation-based inference. Chapters 12, 13, 14, 15, 16, 17, 18, then 43 as a large-scale application.

The function-space view. Chapters 23, 24, 25, 26, 38, 31, 17, and 27. This route never leaves function space, and is the natural one for readers coming from Gaussian processes or spatial statistics.

Priors, model selection, and inductive bias. Chapters 28, 29, 30, 31, 11, and 34. The question running through all six is what we are entitled to assume before seeing data, and how much of it can be learned instead.

Theory and foundations. Chapters 52, 29, 32, 33, 35, 22, 49, and 55. These chapters ask what Bayesian inference means for singular, over-parameterised, non-identifiable models, and what guarantees survive.

Large models and modern systems. Chapters 43, 15, 39, 37, 41, 42, 51, and 47.

Uncertainty for decisions. Chapters 10, 44, 53, 55, and 56. The unifying theme is that a posterior earns its cost only through the decisions it improves: abstention, data acquisition, robustness under ambiguity, and exploration.

BayesAI consortium authors

Claudio Agostinelli

University of Trento
Trento, Italy

Laurence Aitchison

University of Bristol
Bristol, UK

Emanuel Aldea

Université Paris-Saclay
Gif-sur-Yvette, France

Richard Allmendinger

The University of Manchester
Manchester, UK

Sebastian Ament

Meta
New York, USA

Ben Anson

University of Bristol
Bristol, UK

Julyan Arbel

Centre Inria de l'Université Grenoble Alpes
Grenoble, France

Eytan Bakshy

Meta
New York, USA

Max Balandat

Meta
New York, USA

Nicola Bariletto

University of Texas at Austin
Austin, USA

Freddie Bickford Smith

University of Oxford
Oxford, UK

Battista Biggio

University of Cagliari
Cagliari, Italy

Michele Caprio

University of Warwick
Coventry, UK

Neil Chada

City University of Hong Kong

Hong Kong, Hong Kong SAR, China

Wenlin Chen

University of Cambridge
Cambridge, UK

Wenlong Chen

Imperial College London
London, UK

Nathaël Da Costa

University of Tübingen, Tübingen AI Center
Tübingen, Germany

Nico Daheim

TU Darmstadt
Darmstadt, Germany

Andreas Damianou

Spotify
London, UK

Michael Deistler

University of Tübingen, Tübingen AI Center
Tübingen, Germany

David Dunson

Duke University
Durham, USA

Nikita Durasov

École polytechnique fédérale de Lausanne
Lausanne, Switzerland

Maurizio Filippone

King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

Lucia Filippozzi

University of Trento
Trento, Italy

Gergely Flamich

Imperial College London
London, UK

Vincent Fortuin

University of Technology Nuremberg
Nuremberg, Germany

Gianni Franchi

AMIAD
Palaiseau, France

Giorgio Fumera

University of Cagliari
Cagliari, Italy

Yarin Gal

University of Oxford
Oxford, UK

Wenbo Gong

Microsoft
Cambridge, UK

Yuqi Gu

Columbia University
New York, USA

Cheng Guo

The University of Manchester
Manchester, UK

Nick Hauptvogel

University of Copenhagen
Copenhagen, Denmark

Jiajun He

University of Cambridge
Cambridge, UK

Philipp Hennig

University of Tübingen, Tübingen AI Center
Tübingen, Germany

José Miguel Hernández-Lobato

University of Cambridge
Cambridge, UK

Nhat Ho

University of Texas at Austin
Austin, USA

Clara Hoffmann

Karlsruhe Institute of Technology
Karlsruhe, Germany

Aliaksandr Hubin

NMBU and University of Oslo
Oslo, Norway

Christian Igel

University of Copenhagen
Copenhagen, Denmark

Ajay Jasra

Chinese University of Hong Kong, Shenzhen
Shenzhen, China

Matt Jones

U. Colorado, Boulder
Colorado, USA

Nadja Klein

Karlsruhe Institute of Technology
Karlsruhe, Germany

Olivier Laurent

Université Paris-Saclay

Gif-sur-Yvette, France

Kody J. H. Law

The University of Manchester
Manchester, UK

Emanuele Ledda

University of Cagliari
Cagliari, Italy

Bolian Li

Purdue University
West Lafayette, USA

Yingzhen Li

Imperial College London
London, UK

Xinzhu Liang

The University of Manchester
Manchester, UK

Jakob H Macke

ELLIS Institute Tübingen, University of Tübingen,
Tübingen AI Center
Tübingen, Germany

Stephan Mandt

University of California, Irvine
Irvine, USA

Luckeciano Carvalho Melo

University of Oxford
Oxford, UK

Benjamin Kurt Miller

University of Amsterdam
Amsterdam, Netherlands

Edward Milsom

University of Bristol
Bristol, UK

Bruno Mlodozieniec

University of Cambridge
Cambridge, UK

Bálint Mucsányi

University of Tübingen, Tübingen AI Center
Tübingen, Germany

Samuel Müller

Meta
New York, USA

Kevin Murphy

Google DeepMind
Mountain View, USA

Eric Nalisnick

Johns Hopkins University
Baltimore, U.S.A.

Marcos Negre Saura

The University of Manchester
Manchester, UK

Christopher Nemeth

Lancaster University
Lancaster, U.K.

Huy Nguyen

University of Texas at Austin
Austin, USA

Khai Nguyen

Texas A&M University
College Station, USA

Shreyas Padhy

University of Cambridge
Cambridge, UK

Konstantina Palla

Spotify
London, UK

Wei Pan

The University of Manchester
Manchester, UK

Theodore Papamarkou

PolyShape
Athens, Greece

Dhruvesh Patel

UMass Amherst
Amherst, USA

Alina Peluso

Oak Ridge National Laboratory
Oak Ridge, USA

Simon Pepin Lehalleur

University of Duisburg-Essen
Essen, Germany

Konstantinos Pitas

Centre Inria de l'Université Grenoble Alpes
Grenoble, France

Patrick Pynadath

Purdue University
West Lafayette, USA

Tom Rainforth

University of Oxford
Oxford, UK

Maxime Robeys

University of Bristol
Bristol, UK

Fabio Roli

University of Genova
Genova, Italy

Breeshey Roskams-Hieter

University of Edinburgh
Edinburgh, UK

Simone Rossi

EURECOM

Biot, France

Tim G. J. Rudner

University of Toronto
Toronto, Canada

David Rügamer

LMU Munich, Munich Center for Machine Learning
Munich, Germany

Dino Sejdinovic

Adelaide University
Adelaide, Australia

Yevgeny Seldin

University of Copenhagen
Copenhagen, Denmark

Torben Sell

University of Edinburgh
Edinburgh, UK

Louis Sharrock

University College London
London, U.K.

Sumeetpal Singh

University of Wollongong
Wollongong, Australia

Joanna Sliwa

University of Tübingen, Tübingen AI Center
Tübingen, Germany

Emanuel Sommer

LMU Munich, Munich Center for Machine Learning
Munich, Germany

Qifan Song

Purdue University
West Lafayette, USA

Ba-Hien Tran

EURECOM
Biot, France

Richard E. Turner

University of Cambridge
Cambridge, UK

Tycho F. A. van der Ouderaa

Imperial College London
London, UK

Mark van der Wilk

University of Oxford
Oxford, UK

Mariia Vladimirova

Criteo AI Lab
Paris, France

Sara Wade

University of Edinburgh
Edinburgh, UK

Xi Wang

Johns Hopkins University
Baltimore, U.S.A.

Tobias Weber

University of Tübingen, Tübingen AI Center
Tübingen, Germany

Susan Wei

Monash University
Melbourne, Australia

Christopher K. Wikle

University of Missouri
Columbia, MO, USA

Andrew Gordon Wilson

New York University
New York, USA

Lisa Wimmer

LMU Munich
Munich, Germany

Adam Yang

University of Bristol
Bristol, UK

Yibo Yang

Chan Zuckerberg Initiative
Redwood City, USA

Myungsoo Yoo

University of Texas-Austin
Austin, TX, USA

Cheng Zhang

European Laboratory for Learning and Intelligent Systems (ELLIS)
Cambridge, UK

Likun Zhang

University of Missouri
Columbia, MO, USA

Ruqi Zhang

Purdue University
West Lafayette, USA

Yuren Zhou

Duke University
Durham, USA

Editorial leadership and AI disclosure

Editorial leadership

This textbook was created through a large international collaborative effort by the Bayes AI Consortium, bringing together 106 co-authors. The editorial team consisted of Theodore Papamarkou, David Dunson, Vincent Fortuin, Yarin Gal, Philipp Hennig, Konstantina Palla, Tim Rudner, Sara Wade, Andrew Wilson, and Cheng Zhang. Theodore Papamarkou conceived and initiated the project and led its development and coordination.

Use of AI tools

This book is first and foremost a collection of human effort: the work of many authors, reviewers, and editors. We did, however, also make use of AI tools—including GPT and Claude models—to help polish the content, and we believe readers are entitled to a transparent account of where and how.

The technical substance of the book is human. Every chapter was written by its human authors first, and the ideas, derivations, and judgements are theirs. AI assistance entered afterwards, and mostly in the editorial layers.

Cross-chapter review. Before the editorial pass, chapters were reviewed by authors of other chapters. Some reviewers used AI tools to assist their reviews, while every reviewer read the chapter they were assigned at least once. The cross-chapter review was therefore a mixture of AI-assisted and purely human review.

Editorial review. The editorial pass focused on style, cross-referencing, notational consistency, and similar matters of form rather than content. This stage made heavy use of AI tools. The editorial team first agreed, as humans, on the reviewing criteria and recorded them in a shared instruction file; AI tools were then used to generate editorial feedback against those criteria. Because the resulting feedback concerned styling and consistency rather than technical claims, authors were permitted to use AI tools when addressing it, and a subset of them did.

Reader’s guide. The reader’s guide in the front matter is largely AI-assisted: the editorial team supplied the prompts and the structure, and AI tools were used to expand them into the more detailed per-chapter descriptions. The editors reviewed the result.

Responsibility for the content of this book, including any errors that remain, rests with its human authors and editors.

Acknowledgements

Julyan Arbel acknowledges support from ANR-21-JSTM-0001 and ANR-23-IACL-0006 grants.

Battista Biggio acknowledge support from the EU-funded Horizon Europe project ELSA (GA no. 101070617) the projects SERICS (PE00000014) and FAIR (PE00000013) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. Fabio Roli acknowledge support from the project FISA-2023-00128 funded by the MUR program “Fondo italiano per le scienze applicate”.

Michele Caprio (M. C.) would like to thank Kuk Jin Jang for his help with Figure 55.6. In addition, M. C. acknowledges partial funding by the Army Research Office (ARO MURI W911NF2010080).

The authors acknowledge financial support from Imperial College London through an Imperial College Research Fellowship grant awarded to Gergely Flamich.

Vincent Fortuin was supported by a Branco Weiss Fellowship.

Philipp Hennig, Nathaël da Costa, Tobias Weber, Bálint Mucsányi, and Joanna Sliwa gratefully acknowledge financial support by the European Research Council through ERC CoG Action 101123955 ANUBIS; the DFG Cluster of Excellence “Machine Learning - New Perspectives for Science”, EXC 2064/1; the DFG SPP 2298 (Project HE 7114/5-1), and the Carl Zeiss Foundation, (project "Certification and Foundations of Safe Machine Learning Systems in Healthcare"), as well as funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg.

Christian Igel acknowledges support from the Danish National Research Foundation (DNRF) through the Danish Pioneer Centre for AI (P1) and TreeSense, the Center for Remote Sensing and Deep Learning of Global Tree Resources (DNRF192).

Nadja Klein (N. K.) and Clara Hoffmann (C. H.) would like to thank Moussa Kassem Sbeyti for his helpful feedback. C. H. would like to thank Katja Weimer for her valuable comments and suggestions. N. K. and C. H. acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through the grant KL3037/8-1 and the Emmy Noether grant KL 3037/1-1.

Jakob Macke and Michael Deistler were supported by the German Research Foundation (DFG) under Germany’s Excellence Strategy (EXC number 2064/1; 390727645) and SFB 1233 ‘Robust Vision’ (276693517), the European Union (ERC DeepCoMechTome; 101089288), and the Hector Foundation.

Dhruvesh Patel acknowledges support from IBM under IBM Research Collaboration Agreement No. W1668553 and from the National Science Foundation under grant IIS-2106391.

Tom Rainforth was supported by the EPSRC grant EP/Y037200/1.

Freddie Bickford Smith was supported by the EPSRC Probabilistic AI Hub (EP/Y028783/1).

Part I

Sampling methods

Chapter 1

Introduction to sampling

1.1 Introduction

The fundamental objective of Bayesian statistics is to compute the posterior distribution of a random variable $\theta \in \Theta$ given data $\mathcal{D}$. In the context of supervised learning, where the data $\mathcal{D} = \{X, Y\}$ comprises features/inputs X and responses/labels Y , Bayes' formula for the parameter posterior distribution takes the following form

$$p(\theta | \mathcal{D}) = \frac{p(Y | \theta, X)p(\theta)}{\int_{\Theta} p(Y | \theta, X)p(\theta)d\theta}, \quad (1.1)$$

where $p(Y | \theta, X)$ denotes the likelihood of N observations for the target $Y = \{y_1, \dots, y_N\}$ given inputs $X = \{x_1, \dots, x_N\}$, $p(\theta)$ is the density of the prior distribution for θ , and

$$p(\mathcal{D}) = \int_{\Theta} p(Y | \theta, X)p(\theta)d\theta, \quad (1.2)$$

denotes the *model evidence* of the data. Note that, in general the posterior is often only known up-to a normalizing constant, because, aside from simple models, calculating the integral in the model evidence is not feasible for high-dimensional θ nor is it often available in closed form. Therefore, it is more common to work with the unnormalized posterior $p(\theta | \mathcal{D}) \propto p(Y | \theta, X)p(\theta)$.

We note that the ultimate goal in solving (1.1) is estimating the *posterior predictive* for new inputs x^* , i.e. the pushforward of the posterior through the test likelihood:

$$p(y^* | x^*, \theta), \quad \text{for } \theta \sim p(\theta | \mathcal{D}). \quad (1.3)$$

This is a distributed quantity that incorporates both *epistemic and aleatoric uncertainty* (uncertainty caused by a lack of data or knowledge, respectively), but often we content ourselves with its expectation as a point estimator $\hat{p}$:

$$\hat{p}(y^* | x^*, \mathcal{D}) = \mathbb{E}_{\theta | \mathcal{D}} [p(y^* | x^*, \theta)] = \int_{\Theta} p(y^* | x^*, \theta)p(\theta | \mathcal{D})d\theta. \quad (1.4)$$

Simulating or sampling from $p(y^* | x^*, \theta)$, or from $\hat{p}(y^* | x^*)$, or $p(y^* | x^*, \hat{\theta})$ for some point estimator $\hat{\theta}$ is not considered explicitly in this chapter, although it is worth noting that the posterior predictive is often better behaved than the parameter posterior. One primary reason for this may be that it eliminates the non-identifiability arising from exchangeability of the weights on a given layer.

For the remainder of this chapter, we will consider sampling from a generic target distribution $\pi(\theta) = \kappa(\theta)/Z$, where $Z = \int_{\Theta} \kappa(\theta)d\theta$ may or may not be known. The reader can think of $\pi(\theta) := p(\theta | \mathcal{D})$ as the posterior already introduced in (1.1), but where we drop the data $\mathcal{D}$ for notational convenience.

1.2 Monte Carlo

The most fundamental sampling algorithm is the Monte Carlo algorithm [1406], and it is the basis for many other sampling schemes. Monte Carlo works as follows. Suppose we have a random variable $\theta \in \Theta$ from a distribution which is continuous on $\mathbb{R}^P$. Let π be the probability density function for θ and assume that we are able to simulate

independent and identically distributed (i.i.d.) samples from π . Note that we may refer to π as the distribution itself. For some function of interest h , which is integrable with respect to π , we wish to estimate an expectation

$$\mathbb{E}[h(\theta)] = \int h(\theta)\pi(\theta)d\theta, \quad (1.5)$$

which is often not available in closed form. Quadrature methods [375] are suitable in low parameter dimension $P \leq 5$, and with tricks up to $P \approx 20$ [606] or even $P \approx 100$ [219]. However, for large P , Monte Carlo reigns supreme due to its remarkable ability to achieve a mean square error (MSE), defined as $\text{MSE} = \mathbb{E}[(\hat{h}_M - \mathbb{E}[h(\theta)])^2]$, that decays at the canonical rate $\mathcal{O}(1/M)$ in the number of samples M , regardless of dimension or smoothness of the quantity of interest h .

The Monte Carlo method is extremely simple to use. We wish to generate M independent samples $\theta_1, \theta_2, \dots, \theta_M$ as shown in Algorithm 1.

Algorithm 1: Monte Carlo Algorithm

Inputs : target density π

Output: $\{\theta_i\}_{i=1}^M$

for $i = 1$ **to** M **do**

 simulate $\theta_i \sim \pi$ independently

return $\{\theta_i\}_{i=1}^M$

Using the Monte Carlo samples $\{\theta_i\}_{i=1}^M$ from π , we can estimate the expectation as follows

$$\mathbb{E}[h(\theta)] \approx \hat{h}_M := \frac{1}{M} \sum_{i=1}^M h(\theta_i). \quad (1.6)$$

This beautiful equation has remarkable convergence properties, and it is this gold standard that we aim to achieve with other sampling algorithms. Since the samples $\theta_1, \dots, \theta_M$ are i.i.d. from π , and writing $h_i = h(\theta_i)$, we have

$$\mathbb{E}[\hat{h}_M] = \mathbb{E}\left[\frac{1}{M} \sum_{i=1}^M h_i\right] = \frac{1}{M} \sum_{i=1}^M \mathbb{E}[h_i] = \mathbb{E}[h(\theta)]. \quad (1.7)$$

Thus $\hat{h}_M$ is an *unbiased* estimator of the target integral. Moreover, we can increase the accuracy of our estimator by increasing the number of samples M . By the strong law of large numbers, if $\mathbb{E}[h(\theta)] < \infty$, we can guarantee that the Monte Carlo estimator converges almost surely to the true value as the number of samples grows without bound. Formally,

$$\frac{1}{M} \sum_{i=1}^M h(\theta_i) \rightarrow \mathbb{E}[h(\theta)] \quad \text{almost surely as } M \rightarrow \infty. \quad (1.8)$$

That is, with probability one the estimator will eventually get arbitrarily close to the truth. If we invoke the central limit theorem, then we can also quantify how accurate $\hat{h}_M$ is for a given finite M . Define the variance

$$V = \int \{h(\theta) - \mathbb{E}[h(\theta)]\}^2 \pi(\theta) d\theta, \quad (1.9)$$

and assume $V < \infty$. Then as $M \rightarrow \infty$,

$$\sqrt{M} \left(\frac{\hat{h}_M - \mathbb{E}[h(\theta)]}{\sqrt{V}} \right) \xrightarrow{D} \mathcal{N}(0, 1). \quad (1.10)$$

Equivalently, for large M ,

$$\hat{h}_M \sim \mathcal{N}(\mathbb{E}[h(\theta)], V/M), \quad (1.11)$$

so that the standard error of the Monte Carlo estimator is $\sqrt{V/M}$ and its MSE decays at the canonical rate $\mathcal{O}(M^{-1})$ (i.e. root-MSE $\mathcal{O}(M^{-1/2})$). This $M^{-1/2}$ convergence is independent of the dimension P or smoothness of h , and is the benchmark against which all more advanced sampling methods are compared.

Often, a challenge is that we cannot sample from our target probability distribution π directly, and can only evaluate or sample from a non-negative unbiased estimator of κ . Addressing these challenges has led to the development of the algorithms presented in the remainder of this chapter.

1.3 Rejection Sampling

Suppose that $q(\theta)$ is a probability density function from which we *can* sample relatively easily and that $\pi(\theta)$ is the target density from which we would like to sample. Suppose in addition that there exists a constant, $c > 0$, such that

$$\frac{\pi(\theta)}{q(\theta)} \leq c, \quad (1.12)$$

whenever $\pi(\theta) > 0$. Then, we can formulate an algorithm for simulating M samples from $\pi(\theta)$ as given in Algorithm 2.

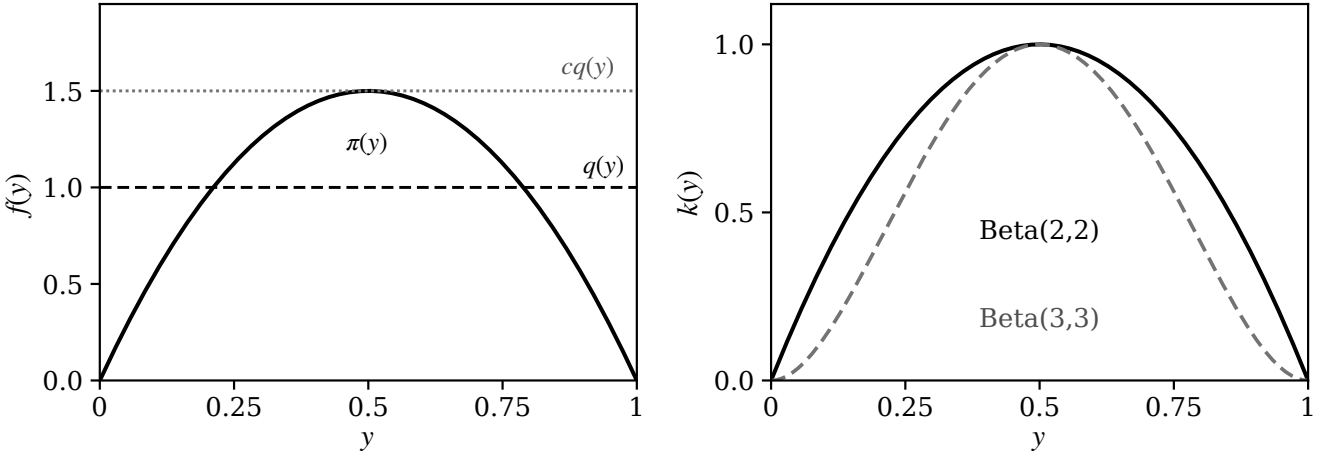


Figure 1.1: Rejection sampling from a Beta(2,2) target $\pi(y)$ using a uniform proposal $q(y)$. Left: q alone does not dominate π , but the envelope cq with $c = 3/2$ does. Right: the acceptance probability $k(y) = \pi(y)/(cq(y))$, shown also for a Beta(3,3) target ($c = 15/8$), for which acceptance is lower.

This algorithm is referred to as *rejection sampling*. We can see that this recipe will be very computationally efficient if the densities $q(\theta)$ and $\pi(\theta)$ are approximately equal and the constant c can be chosen close to 1. In this case, the acceptance probability will generally be close to 1. On the other hand, if $\pi(\theta)$ and $q(\theta)$ are very different, then many runs of the algorithm may be required before a value is finally accepted.

Example 1.1. We illustrate the use of rejection sampling to sample from the Beta(2, 2) distribution. In this case $\pi(\theta) = 6x(1-x)$, $0 \leq x \leq 1$. Suppose that we select $q(\theta)$ to be the uniform density on $(0, 1)$. Then we can see that $\pi/q \leq 3/2$ and the conditions for rejection sampling hold. However, we can see that the acceptance probability will often be small, for example if $\theta < 0.25$ or $\theta > 0.75$ (something that will happen 1/2 of the time) then acceptance happens less than 75% of the time, meaning that we will require significantly more samples than in the ideal i.i.d. case. For the Beta(3, 3) distribution, the rejection rate would be even higher. This can be seen from the right panel of Figure 1.1.

Despite its attraction of ease of implementation and simulation, the efficiency of rejection sampling relies on (i) having a good proposal distribution and (ii) a bound c . Note that it can easily be modified to work with only κ , but these difficulties remain.

1.4 Importance Sampling

The importance sampling estimator of $\mathbb{E}_\pi[h(\theta)]$ begins by re-writing the integral as follows

$$\int_{\Theta} h(\theta)\pi(\theta)d\theta = \int \frac{\pi(\theta)}{q(\theta)}h(\theta)q(\theta)d\theta, \quad (1.13)$$

where q is a density such that $q(\theta) > 0$, whenever $\pi(\theta)h(\theta) \neq 0$. We now produce i.i.d. samples $(\theta_1, \dots, \theta_M)$ from q , and estimate

$$\hat{h} = \frac{1}{M} \sum_{i=1}^M \frac{\pi(\theta_i)}{q(\theta_i)} h(\theta_i) = \frac{1}{M} \sum_{i=1}^M w(\theta_i) h(\theta_i). \quad (1.14)$$

Algorithm 2: Rejection Sampling**Inputs :** target density π , proposal density q , constant $c > 0$ **Output:** $\{\theta_i\}_{i=1}^M$ $i \leftarrow 1$ **while** $i \leq M$ **do** simulate $\theta' \sim q$ and $u \sim U(0, 1)$ **if** $u < \frac{\pi(\theta')}{cq(\theta')}$ **then** $\theta_i \leftarrow \theta'$ $i \leftarrow i + 1$ **return** $\{\theta_i\}_{i=1}^M$

We call this procedure *importance sampling*. The density q is called the proposal or instrumental density and $w(\theta_i) = \frac{\pi(\theta_i)}{q(\theta_i)}$ are the *importance weights*. Note that $\hat{h}$ is an unbiased Monte Carlo estimator of $\mathbb{E}_\pi[h(\theta)]$, as shown in Equation (1.7). There are two reasons why we might be interested in performing importance sampling:

1. Sampling from $\pi(\theta)$ is not possible or too expensive.
2. The function $h(\theta)$ has a large variance, so the conventional unbiased estimator has large Monte Carlo error.

Recalling the results of Section 1.2, the effective test function is now $(\pi/q)h$, and so its regularity dictates the convergence behaviour. Hence, if possible, we should choose a proposal density q such that the variance of $(\pi/q)h$ is small. It can be shown that the optimal proposal distribution is $q \propto \pi|h|$, however that does little good by itself, since we are back to the same problem of approximating q . However, it can provide useful guidance.

Example 1.2. Let $\pi = \mathcal{N}(0, 1)$ and $h = \mathbb{1}(\theta > 4)$, where $\mathbb{1}$ is the indicator function taking the value 1 if the argument is true and 0 otherwise. Since we can simulate i.i.d. from π , it is tempting to do so, i.e. let $\theta_1, \dots, \theta_M \sim \pi$ i.i.d. and set

$$\hat{h} = \frac{1}{M} \sum_{i=1}^M h(\theta_i). \quad (1.15)$$

The number of samples M would have to be very large to get a non-zero estimator. An alternative is to produce an i.i.d. sample from an exponential random variable with rate 1 translated to the right by 4. In this case, we obtain samples $\theta > 4$ such that

$$w(\theta) = \frac{\pi(\theta)}{q(\theta)} = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2} + (\theta - 4)\right). \quad (1.16)$$

Suppose that we only know how to evaluate κ . It is easy to see that an unbiased estimator of the normalizing constant can be built using the following identity

$$Z = \int_{\Theta} \kappa(\theta) d\theta = \int_{\Theta} \frac{\kappa(\theta)}{q(\theta)} q(\theta) d\theta. \quad (1.17)$$

It is left as an exercise to the reader to build a consistent (but biased) self-normalized importance sampling estimator using only κ . Unlike rejection sampling, importance sampling does not waste samples through the rejection step, which could be a high amount depending on the application at hand.

As alluded to already, a major disadvantage of importance sampling is that it can result in high variance. There are numerous methods beyond what has been introduced here to alleviate such issues, including the *sequential importance sampling* and *resampling* methods which will be introduced later in Section 1.6.

1.5 Markov chain Monte Carlo

The preceding sections have introduced Monte Carlo integration and two classical approaches—rejection sampling and importance sampling—for drawing samples from a target distribution $\pi(\theta)$, typically the posterior in Bayesian inference. While these methods are simple to use and theoretically well-founded, they suffer significant limitations in high-dimensional settings. Rejection sampling becomes increasingly inefficient as the dimension grows, due to the difficulty of finding proposal distributions that tightly bound the target. Importance sampling, although

more flexible, tends to suffer from high variance unless the proposal distribution is well matched to the target [7, 261]—a condition that is notoriously hard to satisfy in practice, especially in the multimodal, heavy-tailed, and high-dimensional posteriors encountered in Bayesian deep learning [1292].

Markov Chain Monte Carlo (MCMC) provides a powerful and general framework to overcome these challenges. Rather than attempting to generate independent samples from $\pi(\theta)$, MCMC constructs a *Markov chain* whose stationary distribution is π itself. By simulating a trajectory through parameter space in such a way that, asymptotically, the empirical distribution of visited states converges to the target, MCMC transforms the task of independent sampling into one of designing a suitable transition mechanism that ensures both *ergodicity* and *invariance* with respect to π [1608].

In practical terms, MCMC methods produce a correlated sequence of samples $\{\theta_1, \theta_2, \dots, \theta_M\}$, which, under suitable conditions, can still be used to approximate expectations under π using ergodic averages. The canonical estimator takes the form:

$$\hat{h}_M = \frac{1}{M} \sum_{i=1}^M h(\theta_i), \quad (1.18)$$

which, despite the dependence between samples, converges to the true posterior expectation $\mathbb{E}_\pi[h(\theta)]$ under mild regularity assumptions [613].

In what follows, we introduce several fundamental MCMC algorithms, starting with the *Metropolis–Hastings algorithm*, which provides a generic recipe for constructing π -invariant Markov chains from arbitrary proposal distributions [713, 1169]. We then examine key special cases—random-walk Metropolis, the Metropolis-adjusted Langevin algorithm (MALA), and Hamiltonian Monte Carlo (HMC)—that are particularly relevant in the context of high-dimensional posterior inference in Bayesian deep learning. Throughout, we pay close attention to algorithmic trade-offs, scaling behaviour [509, 1408, 1409], and diagnostic tools for assessing convergence and sampling efficiency [211].

1.5.1 Metropolis–Hastings

Markov chain Monte Carlo (MCMC) proceeds by generating a Markov chain $\theta_1 \rightarrow \theta_2 \rightarrow \dots$ with transition kernel $\mathcal{M}$, described by a density $p(\theta, \theta')$ on Θ^2 , which is invariant with respect to the target posterior π :

$$\int_{\Theta} \pi(\theta') p(\theta', \theta) d\theta' = \pi(\theta). \quad (1.19)$$

The Metropolis–Hastings (MH) algorithm achieves this invariance by first choosing an arbitrary proposal density $q(\theta, \theta')$ that proposes θ' given the current iterate of the Markov chain θ , and then accepting or rejecting each proposed move so that detailed balance—and hence (1.19)—holds.

Concretely, we can define the MH transition kernel as

$$p(\theta, \theta') = q(\theta, \theta') \alpha(\theta, \theta'), \quad (1.20)$$

where the acceptance probability is

$$\alpha(\theta, \theta') = \min \left\{ 1, \frac{\pi(\theta') q(\theta', \theta)}{\pi(\theta) q(\theta, \theta')} \right\}. \quad (1.21)$$

Because $\pi(\theta) \propto \kappa(\theta)$, any unknown normalisation cancels and only the unnormalized density κ is needed. See Algorithm 3 for a full implementation.

Proposition 1.1. *The Metropolis–Hastings algorithm satisfies the equation (1.19).*

Proof. Let us first assume the proposal is accepted. Recalling the definition of $p(\theta, \theta')$ we have

$$p(\theta, \theta') = q(\theta, \theta') \times \alpha(\theta, \theta') = \min \left\{ q(\theta, \theta'), \frac{\pi(\theta') q(\theta', \theta)}{\pi(\theta)} \right\}. \quad (1.22)$$

It follows that

$$\pi(\theta) p(\theta, \theta') = \min \{ \pi(\theta) q(\theta, \theta'), \pi(\theta') q(\theta', \theta) \} = \pi(\theta') p(\theta', \theta), \quad (1.23)$$

by symmetry. The symmetry of the rejection case is left as an exercise. This property is called *detailed balance* and is a sufficient condition for invariance (1.19). $\square$

Algorithm 3: Metropolis-Hastings Algorithm**Inputs :** unnormalized target density κ , proposal distribution q **Output:** $\{\theta_i\}_{i=1}^M$ Generate initial sample $\theta_0 \sim p$, with $\pi \ll p$ (absolute continuity)**for** $i = 0$ **to** $M - 1$ **do** propose new state $\theta' \sim q(\theta_i, \cdot)$

compute the acceptance probability

$$\alpha(\theta_i, \theta') = \min \left\{ 1, \frac{\pi(\theta')q(\theta', \theta_i)}{\pi(\theta_i)q(\theta_i, \theta')} \right\} = \min \left\{ 1, \frac{\kappa(\theta')q(\theta', \theta_i)}{\kappa(\theta_i)q(\theta_i, \theta')} \right\}$$

 draw $u \sim \mathcal{U}[0, 1]$ **if** $u \leq \alpha(\theta_i, \theta')$ **then** $\theta_{i+1} \leftarrow \theta'$ **else** $\theta_{i+1} \leftarrow \theta_i$ **return** $\{\theta_i\}_{i=1}^M$

Invariance with respect to π is a necessary condition for building convergent estimators from a Markov chain. The chain also needs to have certain other stability properties, collectively known as *ergodicity*, to ensure empirical averages (1.6) built from its output converge. A desirable and achievable property is called *geometric ergodicity*, which essentially guarantees linear convergence, i.e. the dependence on the initial condition vanishes geometrically.

Random walk Metropolis

A particularly simple—and historically important—special case of the Metropolis–Hastings algorithm is the *random-walk Metropolis* (RWM) proposal, in which one sets

$$q(\theta, \theta') = q(\theta' - \theta), \quad (1.24)$$

where q is any symmetric density, $q(\theta) = q(-\theta)$. The canonical choice is

$$\theta' = \theta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \lambda^2 I_P), \quad (1.25)$$

so that $\theta' \sim \mathcal{N}(\theta, \lambda^2 I_P)$, i.e. $q(\theta, \cdot) = \mathcal{N}(\theta, \lambda^2 I_P)$. Due to symmetry, the Metropolis acceptance probability simplifies to

$$\alpha(\theta, \theta') = \min \left\{ 1, \frac{\kappa(\theta')}{\kappa(\theta)} \right\}. \quad (1.26)$$

Local moves and tuning. RWM proposals are inherently *local*: each candidate θ' lies in a neighbourhood of θ . If λ is too small, the chain makes only tiny steps and mixes very slowly; if λ is too large, most proposals fall in low-density regions and are rejected. Optimal scaling results show that, for a wide class of P -dimensional targets, the stationary acceptance rate tends to ≈ 0.234 as $P \rightarrow \infty$ when $\lambda \propto P^{-1/2}$ [1408]. In practice, one therefore adapts λ so that the empirical acceptance rate lies between about 20% and 30%.

Preconditioning. When components of θ have very different scales or strong correlations, one can replace $\lambda^2 I_P$ by $\lambda^2 V$, where V is a positive-definite “preconditioning” matrix (often an estimate of the posterior covariance). This *preconditioned RWM* retains symmetry and thus the same simplified acceptance probability, but can yield orders-of-magnitude improvements in computational efficiency, i.e. less samples required to sample the target, because the proposed parameters θ' will be more closely aligned with the posterior distribution, and thus less likely to be rejected.

Gibbs Sampler

Given a multi-variate $\theta = (\theta_1, \theta_2 \dots \theta_n)$, it may be intractable to sample all at once. However if the full conditional distributions for each coordinate are known, then it becomes possible to apply *Gibbs sampling*.

The Gibbs sampling algorithm is fairly straightforward. Given the proposals are known, the Gibbs algorithm updates the sample by updating each coordinate one at a time, i.e. $\theta'_i \sim P(\theta_i \mid \theta_{-i})$, where θ_{-i} denotes all coordinates except i . While it may be tempting to update all coordinates at the same time, this would not properly preserve the joint distribution as all coordinates would be updated independently. We include a simple formulation in Algorithm 4.

Algorithm 4: Gibbs Sampler

Inputs : initial state $\theta = (\theta_1, \theta_2 \dots \theta_n)$; full conditional distributions $P(\theta_i \mid \theta_{-i})$ for $i \in \{1, \dots, n\}$

Outputs: sample set $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$

for each iteration do

for $i \in \{1, \dots, n\}$ **do**

$\theta'_i \sim P(\theta_i \mid \theta_{-i})$

$\theta \leftarrow (\theta_1, \theta_2, \dots, \theta'_i, \dots, \theta_n)$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\theta\}$

Given that the Gibbs sampler is drawing from the correct proposal, the typical Metropolis-Hastings acceptance step is not needed in this case. However, this sampler suffers from slow and highly correlated samples due to the nature of the single coordinate update approach.

It is possible to extend the initial Gibbs sampler to a block-style approach in the case where the joint distribution over multiple coordinates is known. Instead of the single-coordinate marginal, if we know the joint distribution over one-set of coordinates conditioned on another set, it is possible to quickly sample by alternative sampling from each joint distribution. This is known as the Block-Gibbs sampler. More formally, if we have two sets of coordinates I, J , the block Gibbs sampler proceeds by sampling $\theta'_I \sim P(\theta_I \mid \theta_J)$, and then sampling $\theta'_J \sim P(\theta_J \mid \theta'_I)$. This sampling algorithm avoids the pitfalls of the typical Gibbs sampler. However, it is only possible to use this in cases where the conditional distributions over blocks of coordinates is known.

Metropolis-adjusted Langevin Algorithm

The random-walk Metropolis ignores gradient information about π . The Metropolis-Adjusted Langevin Algorithm (MALA) rectifies this by using an Euler-Maruyama discretisation of the overdamped Langevin diffusion

$$d\theta_t = \frac{1}{2} \nabla \log \pi(\theta_t) dt + dW_t, \quad (1.27)$$

where $\{W_t\}_{t \geq 0}$ is standard Brownian motion on $\mathbb{R}^P$, and whose stationary law is π . Concretely, the MALA proposal is

$$\theta' | \theta \sim \mathcal{N} \left(\theta + \frac{\lambda^2}{2} \nabla \log \kappa(\theta), \lambda^2 I_P \right). \quad (1.28)$$

This drift term $\nabla \log \kappa(\theta)$ biases proposed moves towards regions of higher posterior density, enabling larger step-sizes than RWM and better acceptance rates. The MALA proposal also forms the basis for many other wonderful Monte Carlo algorithms, including the stochastic gradient Langevin dynamics (SGLD) [1737] and all other stochastic gradient MCMC (SG-MCMC) algorithms, which will be the focus of the next chapter.

Gradient-based tuning. Under regularity conditions, choosing $\lambda \propto P^{-1/6}$ yields an optimal stationary acceptance rate of approximately 0.574 as $P \rightarrow \infty$ [1409]. In applications, one often adapts λ to target an acceptance rate of about 50% to 60%. This can be done by monitoring the acceptance rate and adjusting λ accordingly, or by using more sophisticated adaptive schemes that adjust the step size based on the empirical gradient information.

Preconditioned MALA and robustness. Analogous to RWM, one can introduce a symmetric preconditioning matrix V , replacing I_P by V both in the covariance and in the drift:

$$\theta' \sim \mathcal{N} \left(\theta + \frac{\lambda^2}{2} V \nabla \log \kappa(\theta), \lambda^2 V \right). \quad (1.29)$$

Preconditioning often dramatically accelerates convergence on anisotropic posteriors. However, because MALA relies on $\nabla \log \kappa$, it can be sensitive to regions where the gradient is large or ill-behaved (e.g. lighter-than-Gaussian tails). Common mitigations include gradient clipping or fallback to RWM outside of a “trust region.”

Monitoring Performance

Assessing the quality of an MCMC run is essential to ensure that empirical averages $\hat{h}_M = \frac{1}{M} \sum_{i=1}^M h(\theta_i)$ reliably approximates $\mathbb{E}_\pi[h(\theta)]$. Three widely used diagnostics are:

1. Autocorrelation and Integrated Autocorrelation Time. For a scalar summary of $h(\theta)$, define the lag- k autocorrelation

$$\rho_k = \frac{\text{Cov}[h(\theta_i), h(\theta_{i+k})]}{\text{Var}_\pi[h(\theta)]}. \quad (1.30)$$

As the Markov chain is stationary, $\rho_0 = 1$, and $\rho_k \rightarrow 0$ as $k \rightarrow \infty$. The slower ρ_k decays, the more correlated is the chain. One can show that, via the Markov-chain central limit theorem,

$$\sqrt{M}(\hat{h}_M - \mathbb{E}_\pi[h]) \xrightarrow{\mathcal{D}} \mathcal{N}(0, V_{\text{eff}}), \quad \text{where} \quad V_{\text{eff}} = \text{Var}_\pi[h] \left(1 + 2 \sum_{k=1}^{\infty} \rho_k\right). \quad (1.31)$$

The quantity $\tau_{\text{int}} = 1 + 2 \sum_{k=1}^{\infty} \rho_k$ is the *integrated autocorrelation time*, and the *effective sample size* $\text{ESS} = M/\tau_{\text{int}}$ measures the number of independent draws represented by M correlated samples. Figure 1.3 illustrates autocorrelation decay. A related metric is the *expected squared jumping distance* (ESJD) $\mathbb{E}[\|\theta_{i+1} - \theta_i\|^2]$, which balances move size against acceptance rate. Larger moves will reduce the correlation between elements of the Markov chain, however, larger moves are also less likely to be accepted, which means that the chain does not evolve. Therefore, the challenge for practitioners, is to balance between small moves (high correlation and high acceptance rate) and large moves (low correlation and low acceptance rate).

2. Trace Plots. Plotting the sequence $\{h(\theta_i)\}_{i=1}^M$ versus iteration i provides a simple-to-interpret visual representation of mixing and stationarity. Well-mixed chains rapidly traverse the bulk of the posterior, showing no apparent trends or “stickiness” (see Left Panel Figure 1.2). Conversely, slow mixing or multimodality is signalled by long periods of drift or entrapment in subregions of the posterior (see Right Panel Figure 1.2). Trace plots are a useful visual tool for diagnosing the efficiency of an MCMC algorithm, however, they tend to be most useful for low-dimensional posteriors as it is easier to visualise all of the parameters θ , whereas, for high-dimensional posteriors, visualising all of the parameters in the Markov chain individually is often impractical.

3. Potential Scale Reduction (Gelman–Rubin) Diagnostic. When multiple chains $\{\theta_i^{(c)}\}$ are run from overdispersed starting points, one can compare the *between-chain variance* B to the *within-chain variance* W . The *potential scale reduction factor*

$$\hat{R} = \sqrt{\frac{\frac{M-1}{M}W + \frac{1}{M}B}{W}} \quad (1.32)$$

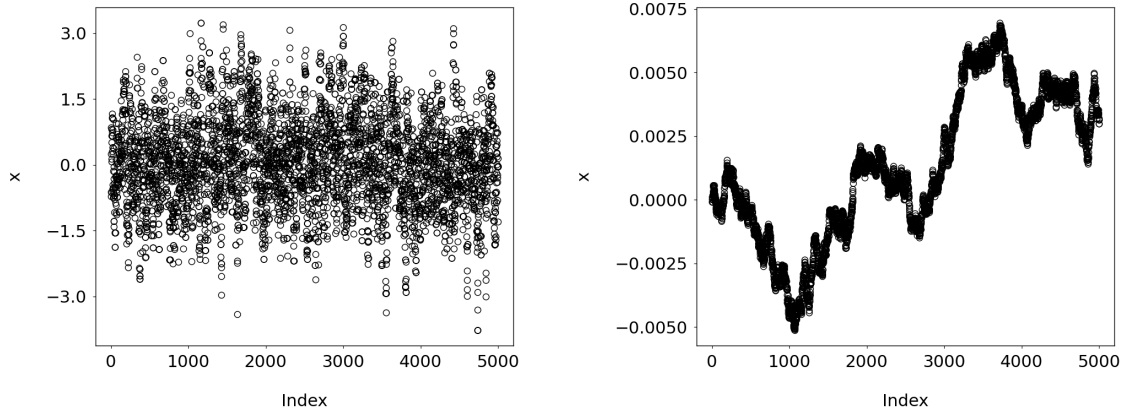
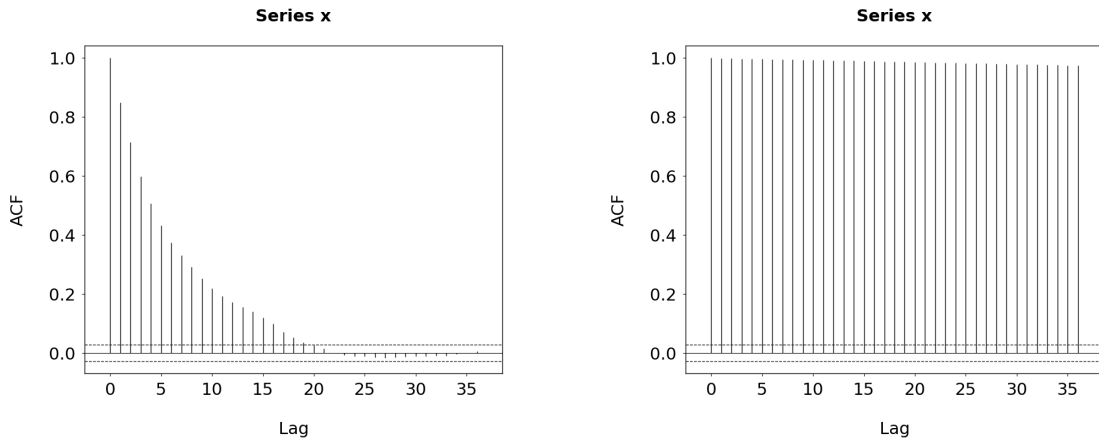
tends to 1 as all chains converge to the same target distribution [600]. Values of $\hat{R} < 1.1$ are commonly taken to indicate satisfactory convergence, after which samples from all chains may be pooled to produce an approximation to the posterior π .

In practice, one would usually employ a combination of these diagnostics: trace plots for qualitative assurance, autocorrelation and ESS for quantitative accuracy, and $\hat{R}$ to guard against failure to explore multimodal targets.

Example 1.3. We present a Metropolis-Hastings experiment aiming to sample from a target $\pi(\theta) = N(0, 1)$. Our proposal is based on a random walk scheme (Section 1.5.1) $\theta' = \theta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \lambda^2 I)$ is additive Gaussian noise. We run two different experiments, with different choices of step size parameter $\lambda \in \{0.01, 0.9\}$. Our simulations are plotted below in Figures 1.2 and 1.3. For the larger choice of λ , we notice that the chain explores the space well as the iterations progress, with the ACF decorrelating rapidly. However, for the smaller value of λ , the chain wanders slowly and mixes poorly, and the ACF decreases much slower, remaining close to one at moderate lags.

1.5.2 Hamiltonian Monte Carlo

Random-walk Metropolis (RWM) requires proposal steps of size $\lambda = O(d^{-1/2})$ to maintain a non-degenerate acceptance rate in dimension d , whereas MALA can take steps of size $O(d^{-1/6})$. *Hamiltonian Monte Carlo* (HMC) goes a step further: with an integration step size $\epsilon = O(d^{-1/4})$ it still enjoys an acceptance rate bounded away from 0 while exploring much larger regions of parameter space between consecutive accept-reject decisions [140, 315, 456, 1229]. The key is to embed the d -dimensional parameter θ in a $2d$ -dimensional *phase space* and simulate approximate Hamiltonian dynamics that preserve a surrogate “total energy”.

Figure 1.2: Trace plots for $\lambda = 0.9$ (left) and $\lambda = 0.01$ (right).Figure 1.3: Corresponding ACFs for $\lambda = 0.9$ (left) and $\lambda = 0.01$ (right).

Extended target and Hamiltonian. Introduce an auxiliary *momentum* variable $z \in \mathbb{R}^d$ and a positive-definite *mass matrix* M_0 . Define the Hamiltonian

$$H(\theta, z) = U(\theta) + K(z) := -\log \pi(\theta) + \frac{1}{2} z^\top M_0^{-1} z, \quad (1.33)$$

so that the joint density $\tilde{\pi}(\theta, z) \propto \exp[-H(\theta, z)]$ has marginal $\pi(\theta)$ and independent $z \sim \mathcal{N}(0, M_0)$.

Hamiltonian dynamics. The continuous equations of motion are

$$\frac{d\theta}{dt} = M_0^{-1} z, \quad \frac{dz}{dt} = \nabla_\theta \log \pi(\theta), \quad (1.34)$$

which conserve H exactly and therefore move along surfaces of constant joint density. Exact simulation is impossible for general targets π , so HMC uses the second-order, symplectic *leapfrog* integrator [1011] with step

size Δt :

$$\begin{aligned} z_{t+\frac{1}{2}\Delta t} &= z_t + \frac{\Delta t}{2} \nabla_{\theta} \log \pi(\theta_t), \\ \theta_{t+\Delta t} &= \theta_t + \Delta t M_0^{-1} z_{t+\frac{1}{2}\Delta t}, \\ z_{t+\Delta t} &= z_{t+\frac{1}{2}\Delta t} + \frac{\Delta t}{2} \nabla_{\theta} \log \pi(\theta_{t+\Delta t}). \end{aligned} \tag{1.35}$$

One repeats this map L times, with $\epsilon = \Delta t$ and $T = L\epsilon$ is the *integration time*.

HMC kernel. Starting from θ_{i-1} :

1. Draw momentum $z \sim \mathcal{N}(0, M_0)$.
2. Apply L leapfrog steps (1.35) to obtain (θ', z') .
3. Accept (θ', z') with probability

$$\alpha = \min \left\{ 1, \exp [H(\theta_{i-1}, z) - H(\theta', z')] \right\}. \tag{1.36}$$

4. Set $\theta_i = \theta'$ if accepted, otherwise $\theta_i = \theta_{i-1}$; discard z' .

Because the leapfrog map is volume-preserving (i.e. the Jacobian = 1) and symmetric (its inverse is obtained by $(\theta, z) \mapsto (\theta, -z)$ and running the L steps backward), the Metropolis correction guarantees $\tilde{\pi}$ is invariant; the posterior π is therefore stationary for the marginal chain on θ .

Algorithm 5: Hamiltonian Monte Carlo Kernel

Inputs : initial state θ_0 , target distribution π , mass matrix M_0

Output: $\{\theta_i\}_{i=1}^M$

for $i = 1$ **to** M **do**

generate initial momentum $z \sim \mathcal{N}(0, M_0)$

for $l = 1$ **to** L **do**

generate $\theta_{l\Delta t}$ and $z_{l\Delta t}$ from (1.35) applied at $(\theta_{(l-1)\Delta t}, z_{(l-1)\Delta t})$, with $\theta_0 = \theta_{i-1}$ and $z_0 = z$

let $(\theta', z') \leftarrow (\theta_{L\Delta t}, z_{L\Delta t})$ and generate $u \sim U[0, 1]$

if $u \leq \min \{1, \exp [H(\theta, z) - H(\theta', z')]\}$ **then**

$\theta_i \leftarrow \theta'$

else

$\theta_i \leftarrow \theta_{i-1}$

return $\{\theta_i\}_{i=1}^M$

High-dimensional scaling. For product targets $\pi(\theta) = \prod_{i=1}^d f(\theta^i)$ with $M_0 = I_d$, one can show [509] that to keep a well-behaved acceptance rate, one must scale $\epsilon = O(d^{-1/4})$, hence $L = O(d^{1/4})$. The computational cost per effectively independent draw is therefore $O(d^{1/4})$, compared with $O(d^{1/3})$ for MALA and $O(d)$ for RWM.

Practical tuning.

- **Step size ϵ .** During warm-up, adapt ϵ (e.g. with dual-averaging) to target an acceptance rate in the 60–80% range; theory suggests 65% is asymptotically optimal [140].
- **Trajectory length $T = L\epsilon$.** Too small T gives RWM-like behaviour; too large wastes computation and may return to near the starting point. Popular heuristics randomise L or employ the No-U-Turn criterion [762].
- **Mass matrix M_0 .** Setting M_0 to an estimate of the posterior covariance (or its diagonal) acts like the preconditioning matrices used with RWM and MALA, dramatically improving convergence when parameters are on different scales.

Because HMC combines long, momentum-driven proposals with high acceptance probabilities, it has become the default sampler in many Bayesian deep-learning applications, where posteriors are high-dimensional, strongly correlated and feature narrow valleys that thwart simpler MCMC methods.

1.6 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods are a set of Monte Carlo algorithms designed to sample from a sequence of target probability densities $\{\pi_n(\theta_{1:n})\}$ of increasing dimension, where each one is defined on the product space $\Theta^n = \prod_{i=1}^n \Theta$ [401, 447]. This is referred to as the *smoothing distribution* in the context of state-space models. Often one may be interested only in the time n marginal $\pi_n(\theta_n)$, which is known as the *filtering distribution*. Suppose

$$\pi_n(\theta_{1:n}) = \frac{\kappa_n(\theta_{1:n})}{Z_n}, \quad (1.37)$$

where the un-normalized density admits the decomposition

$$\kappa_n(\theta_{1:n}) = \kappa_n(\theta_n | \theta_{1:n-1}) \kappa_{n-1}(\theta_{1:n-1}), \quad (1.38)$$

and $\kappa_n(\theta_n | \theta_{1:n-1})$ and $\kappa_1(\theta_1)$ can be evaluated, and Z_n is the unknown normalizing constant given by

$$Z_n = \int \kappa_n(\theta_{1:n}) d\theta_{1:n}. \quad (1.39)$$

Note that we write

$$\int d\theta_{1:n} \quad \text{which is equivalent to} \quad \int_{\Theta} \int_{\Theta} \cdots \int_{\Theta} d\theta_1 d\theta_2 \cdots d\theta_n.$$

SMC provides an approximation of $\pi_1(\theta_1)$ and an estimate of Z_1 at time 1 then an approximation of $\pi_2(\theta_{1:2})$ and an estimate of Z_2 at time 2 and so on. The approximations are done sequentially in time:

- At time step 1, one samples N samples $\{\theta_1^{(i)}\}_{i=1}^N$ from some given function $q_1(\theta_1)$ and then approximates $\pi_1(\theta_1)$ and Z_1 using these samples;
- At time step 2, sample $\{\theta_2^{(i)}\}_{i=1}^N$ from some function $q_2(\theta_2 | \theta_1)$, and use the cumulative samples to approximate $\pi_2(\theta_{1:2})$ and Z_2 .

Consider another probability density $q_n(\theta_{1:n}) = q_n(\theta_n | \theta_{1:n-1}) q_{n-1}(\theta_{1:n-1})$ that is easy to sample from, and such that the support of $q_n(\theta_{1:n})$ includes the support of $\pi_n(\theta_{1:n})$. Then, we have the following importance sampling (IS) identities

$$\pi_n(\theta_{1:n}) = \frac{\kappa_n(\theta_{1:n})}{Z_n} = \frac{w_n(\theta_{1:n}) q_n(\theta_{1:n})}{Z_n}, \quad (1.40)$$

$$Z_n = \int w_n(\theta_{1:n}) q_n(\theta_{1:n}) d\theta_{1:n}, \quad (1.41)$$

where $w_n(\theta_{1:n})$ is the unnormalized weight function and in light of (1.38) we have the following recursive definition of the weights

$$w_n(\theta_{1:n}) = \frac{\kappa_n(\theta_{1:n})}{q_n(\theta_{1:n})} = \frac{\kappa_n(\theta_n | \theta_{1:n-1})}{q_n(\theta_n | \theta_{1:n-1})} w_{n-1}(\theta_{1:n-1}) =: \alpha(\theta_n | \theta_{1:n-1}) w_{n-1}(\theta_{1:n-1}). \quad (1.42)$$

Assume we draw, sequentially in time, M independent samples $\theta_{1:n}^{(i)}$, $i = 1, \dots, M$, from $q_n(\theta_{1:n})$ (see Algorithm 6). We have the following unbiased estimator of Z_n :

$$\hat{Z}_n = \frac{1}{M} \sum_{i=1}^M w_n(\theta_{1:n}^{(i)}). \quad (1.43)$$

A consistent self-normalized importance sampling estimator of $\mathbb{E}_n(h_n)$, for an arbitrary function $h_n : \Theta^n \rightarrow \mathbb{R}$, is given by

$$\widehat{\mathbb{E}}_n(h_n) = \frac{\sum_{i=1}^N w_n(\theta_{1:n}^{(i)}) h_n(\theta_{1:n}^{(i)})}{\sum_{i=1}^N w_n(\theta_{1:n}^{(i)})} = \sum_{i=1}^M W_n^{(i)} h_n(\theta_{1:n}^{(i)}),$$

where

$$W_n^{(i)} = \frac{w_n(\theta_{1:n}^{(i)})}{\sum_{i=1}^M w_n(\theta_{1:n}^{(i)})}. \quad (1.44)$$

Algorithm 6: Sequential Importance Sampling (SIS) to estimate $\mathbb{E}_n(\theta_n)$ for $n \geq 1$

Inputs : number of particles M , proposal distributions q_n , adjustment multipliers α_n
Outputs: estimates $\hat{\theta}_n$ and $\hat{Z}_n$ for $n \geq 1$
for $i = 1$ **to** M **do**

 initialize weights $w_1^{(i)} \leftarrow 1/M$
 $n \leftarrow 1$
repeat

 for $i = 1$ **to** M **do**

 sample $\theta_n^{(i)} \sim q_n(\theta_n \mid \theta_{1:n-1}^{(i)})$

update unnormalized weight

$$w_n(\theta_{1:n}^{(i)}) \leftarrow \alpha_n(\theta_n^{(i)} \mid \theta_{1:n-1}^{(i)}) w_{n-1}(\theta_{1:n-1}^{(i)})$$

normalize weight

$$W_n^{(i)} \leftarrow \frac{w_n(\theta_{1:n}^{(i)})}{\sum_{j=1}^M w_n(\theta_{1:n}^{(j)})}$$

compute estimates

$$\hat{\theta}_n \leftarrow \sum_{i=1}^M W_n^{(i)} \theta_n^{(i)}, \quad \hat{Z}_n \leftarrow \frac{1}{M} \sum_{i=1}^M w_n(\theta_{1:n}^{(i)})$$

 $n \leftarrow n + 1$
until termination criterion met

These are the ingredients of the *sequential importance sampling* (SIS) algorithm, given in Algorithm 6. In Algorithms 6 and 7 we take $h_n(\theta_{1:n}) = \theta_n$.

As n grows the weights will degenerate, in the sense that one will dominate the rest. This can be monitored by the effective sample size, which is related to the variance of the weights:

$$\text{ESS} = \frac{\left(\sum_{i=1}^M w_n^{(i)}\right)^2}{\sum_{i=1}^M (w_n^{(i)})^2}. \quad (1.45)$$

One way to mitigate this somewhat is by resampling, i.e. for $i = 1, \dots, M$, let

$$\theta_{1:n}^{(i)} \leftarrow \theta_{1:n}^{(j)}, \quad \text{for } j \sim (W_n^{(1)}, \dots, W_n^{(M)}), \quad \text{and } w_n^{(i)} \leftarrow 1/M. \quad (1.46)$$

Resampling slightly increases the variance of the estimator, but the particles are rejuvenated at the filtering end, and under suitable conditions the filter can even remain stable online for infinite time [401]. Note that this branching of the particles forward in time means coalescence backward in time, so the particles still degenerate for small times, which is problematic for the smoother. SIS along with resampling is called sequential importance resampling (SIR) or sequential Monte Carlo (SMC). See Algorithm 7. In the context of a state space model, q_n is often chosen as the forward evolution kernel of the hidden process, and in the filtering context this specific choice is often called the *bootstrap particle filter*. In practice, one resamples when the ESS is less than a threshold (e.g. $M/2$ or $M/4$). See Chopin and Papaspiliopoulos [297] for a recent comprehensive introduction.

Algorithm 7: Sequential Monte Carlo (SMC) to estimate $\mathbb{E}(\theta_n)$ for $n \geq 1$

Inputs : number of particles M , proposals q_n , adjustment multipliers α_n
Outputs: estimates $\hat{\theta}_n$ and $\hat{Z}_n$ for $n \geq 1$

// Initialization at $n = 1$
for $i = 1$ **to** M **do**

 sample $\theta_1^{(i)} \sim q_1(\theta_1)$

 compute $w_1(\theta_1^{(i)}) \leftarrow \kappa_1(\theta_1^{(i)})/q_1(\theta_1^{(i)})$

$$W_1^{(i)} \leftarrow \frac{w_1(\theta_1^{(i)})}{\sum_{j=1}^M w_1(\theta_1^{(j)})} \quad \text{for } i = 1, \dots, M, \quad \hat{Z} \leftarrow 1$$

// Iterate for subsequent times

 $n \leftarrow 1$
repeat

 // Effective sample size at time n

$$ESS_n \leftarrow \left(\sum_{i=1}^M \left(W_n^{(i)} \right)^2 \right)^{-1}$$

 // Optional resampling at time n

 if $ESS_n \leq M/2$ **then**

 // Multinomial resampling with probabilities $W_n^{(1:M)}$

 for $i = 1$ **to** M **do**

 draw $j \sim \text{Categorical}(W_n^{(1)}, \dots, W_n^{(M)})$

 $\theta_{1:n}^{(i)} \leftarrow \theta_{1:n}^{(j)}$

 $w_n^{(i)} \leftarrow 1/M$

$$\hat{Z} \leftarrow \left(\frac{1}{M} \sum_{i=1}^M w_n(\theta_{1:n}^{(i)}) \right) \hat{Z}$$

 // Advance to time $n + 1$

 $n \leftarrow n + 1$

 for $i = 1$ **to** M **do**

 sample $\theta_n^{(i)} \sim q_n(\theta_n \mid \theta_{1:n-1}^{(i)})$

update unnormalized weight

$$w_n(\theta_{1:n}^{(i)}) \leftarrow \alpha_n(\theta_n^{(i)} \mid \theta_{1:n-1}^{(i)}) w_{n-1}(\theta_{1:n-1}^{(i)})$$

$$W_n^{(i)} \leftarrow \frac{w_n(\theta_{1:n}^{(i)})}{\sum_{j=1}^M w_n(\theta_{1:n}^{(j)})} \quad \text{for } i = 1, \dots, M$$

 // Estimates at time n

$$\hat{\theta}_n \leftarrow \sum_{i=1}^M W_n^{(i)} \theta_n^{(i)}, \quad \hat{Z}_n \leftarrow \frac{1}{M} \sum_{i=1}^M w_n(\theta_{1:n}^{(i)}) \hat{Z}$$

until *termination criterion met*

1.7 Summary

This chapter introduced the Monte Carlo algorithm and a number of methods that develop it. Each addresses the same underlying difficulty: simulating from, or estimating expectations under, a target distribution $\pi(\theta)$ that in most applications can only be evaluated up to an unknown normalising constant.

When independent samples from π are available, Monte Carlo integration (Section 1.2) applies directly. When

they are not, rejection sampling (Section 1.3) and importance sampling construct estimators from an accessible proposal distribution. Both are exact in principle, but become inefficient in high dimension unless the proposal closely matches the target.

Markov chain Monte Carlo (Section 1.5) removes the need for independent draws by constructing a Markov chain whose stationary distribution is the target. The Metropolis–Hastings algorithm (Section 1.5.1) provides the general recipe, with the random-walk Metropolis algorithm (Section 1.5.1), the Gibbs sampler, the Metropolis-adjusted Langevin algorithm (Section 1.5.1), and Hamiltonian Monte Carlo (Section 1.5.2) as the special cases treated here. Since the samples are correlated, step-size tuning and the convergence diagnostics of Section 1.5.1 are essential in practice.

Sequential Monte Carlo (Section 1.6) extends importance sampling to a sequence of targets on nested product spaces, as in filtering. Sequential importance sampling with resampling propagates weighted particles forward in time, with resampling when the effective sample size drops.

The methods play complementary roles: rejection and importance sampling suit low-dimensional or well-matched problems, MCMC is the standard tool for high-dimensional Bayesian computation, and sequential Monte Carlo is suited to inference along a sequence. These ideas are taken further in the chapters that follow. For example, stochastic gradient MCMC is covered in Chapter 2, and sequential Monte Carlo *samplers*, which applies SMC to an artificial sequence of intermediate targets with MCMC mutations, with tempering and multilevel extensions for neural network posteriors, in Chapter 4.

Chapter 2

Stochastic Gradient MCMC

Deep learning models extensively use stochastic gradients (i.e., mini-batches) to reduce the memory requirements of model training [192, 809]. Under suitable assumptions, stochastic gradients computed from individual samples or minibatches can be used in place of full-batch gradients while retaining almost-sure convergence guarantees. [1482]. In this chapter, we introduce stochastic gradient MCMC (SG-MCMC) methods, in which stochastic gradients make MCMC suitable for modern deep learning models.

2.1 Introduction

Standard MCMC algorithms are asymptotically exact samplers under suitable conditions [713]; Chapter 1 provides an introduction to MCMC and HMC. For Langevin-based methods, an appropriate decaying step-size schedule yields asymptotic convergence to the target distribution [266]. However, using the entire dataset to compute each update makes full-batch MCMC impractical for deep learning models, which commonly have tens of millions of parameters. Inspired by stochastic gradient descent (SGD), stochastic gradient MCMC (SG-MCMC) uses stochastic gradients to approximate full-batch MCMC updates. Similar ideas have been transferred to mini-batch Metropolis–Hastings (MH) algorithms [917, 1357, 1853]. However, mini-batch MH often relies on strong assumptions, such as bounded log-likelihoods or bounded gradients, which are not guaranteed for deep learning models. SG-MCMC [273, 430, 1737] addresses these practical concerns under fewer assumptions and with substantially lower memory requirements, thereby bridging the gap between MCMC and Bayesian neural networks (BNNs).

2.2 Classic SG-MCMC Algorithms

In this section, we introduce two concrete examples of SG-MCMC algorithms that are widely used for deep learning models.

2.2.1 Notation

Let θ be the parameters of a deep learning model, and let $\mathcal{S} = \{\theta_s\}_{s=1}^M$ be a sample set collected during the MCMC process. Due to memory constraints, we can collect only a finite number of samples to characterise the posterior distribution:

$$p(\theta \mid \mathcal{D}) \approx \frac{1}{M} \sum_{s=1}^M \delta(\theta - \theta_s), \quad (2.1)$$

where $\delta(\cdot)$ is the Dirac delta function (also known as the unit impulse) [839]. At inference time, the sample set is used for Bayesian model averaging (BMA) [279], in which the predictive probabilities of all collected samples contribute to the final prediction:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D})} [p(y \mid x, \theta)] \approx \arg \max_{y \in \mathcal{Y}} \frac{1}{M} \sum_{s=1}^M p(y \mid x, \theta_s). \quad (2.2)$$

The goal of each SG-MCMC algorithm is to sample from the posterior distribution and collect such a sample set to reconstruct an approximate posterior distribution.

2.2.2 Stochastic Gradient Langevin Dynamics

Stochastic Gradient Langevin Dynamics (SGLD) [1737] is the first well-known SG-MCMC algorithm. The full-batch energy function is $U(\theta) = -\sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \log p(\mathbf{y} | \mathbf{x}, \theta) - \log p(\theta)$. Its mini-batch estimate is

$$\tilde{U}(\theta) = -\frac{|\mathcal{D}|}{|\Xi|} \sum_{(\mathbf{x}, \mathbf{y}) \in \Xi} \log p(\mathbf{y} | \mathbf{x}, \theta) - \log p(\theta), \quad (2.3)$$

where $\Xi \subset \mathcal{D}$ is the sampled batch. The sampling process for SGLD follows the Langevin dynamics:

$$\theta_{t+1} \leftarrow \theta_t - \alpha_t \nabla_{\theta_t} \tilde{U}(\theta_t) + \sqrt{2\alpha_t} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (2.4)$$

where α_t is the step size at iteration t . The complete SGLD framework is outlined in Algorithm 8. Compared with standard SGD, SGLD introduces an additional random-noise term $\sqrt{2\alpha_t} \epsilon_t$. As noted by Welling and Teh [1737], SGLD approaches standard unadjusted Langevin dynamics [1410] when the step size α_t is sufficiently small, as the injected noise dominates the stochastic-gradient noise. Under standard regularity conditions, convergence is guaranteed by a decaying step-size schedule [193, 1097, 1737]. Any schedule satisfying i) $\sum_{t=1}^{\infty} \alpha_t = \infty$ and ii) $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ allows the sampler to explore the target distribution while controlling its asymptotic error [279]. Notably, mini-batch noise in SGD without externally injected noise can produce behaviour qualitatively similar to SGLD, but converges to a biased approximate posterior [1128].

Algorithm 8: Stochastic Gradient Langevin Dynamics (SGLD)

Inputs : dataset $\mathcal{D}$, initial sample $\theta_0 \in \Theta$, step-size schedule $\{\alpha_t\}_{t \geq 0}$
Outputs: collected samples $\mathcal{S} \subset \Theta$
 $\theta \leftarrow \theta_0$; $\mathcal{S} \leftarrow \emptyset$
for each iteration do
 $\Xi \leftarrow$ a mini-batch sampled from $\mathcal{D}$
 $\tilde{U} \leftarrow -\frac{|\mathcal{D}|}{|\Xi|} \log p(\Xi | \theta) - \log p(\theta)$ // Compute energy
 $\theta \leftarrow \theta - \alpha_t \nabla_{\theta} \tilde{U} + \sqrt{2\alpha_t} \epsilon$ // $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $\mathcal{S} \leftarrow \mathcal{S} \cup \{\theta\}$

2.2.3 Stochastic Gradient Hamiltonian Monte Carlo

Full-batch Hamiltonian Monte Carlo (HMC) has achieved gold-standard performance in Bayesian inference [826]. HMC incorporates a kinetic-energy term characterised by a set of auxiliary “momentum” variables. To adapt HMC to deep learning models and reduce its memory requirements, stochastic gradient HMC (SG-HMC) [273] eliminates the need for full-batch gradients and removes the MH correction used by full-batch HMC. Specifically, naive SG-HMC replaces the full-batch energy in the Hamiltonian with its mini-batch estimate:

$$\tilde{H}(\theta, \mathbf{r}) = \tilde{U}(\theta) + \frac{1}{2} \mathbf{r}^\top \mathbf{M}^{-1} \mathbf{r} = -\frac{|\mathcal{D}|}{|\Xi|} \sum_{(\mathbf{x}, \mathbf{y}) \in \Xi} \log p(\mathbf{y} | \mathbf{x}, \theta) - \log p(\theta) + \frac{1}{2} \mathbf{r}^\top \mathbf{M}^{-1} \mathbf{r}, \quad (2.5)$$

where $\mathbf{r}$ is the momentum and $\mathbf{M}$ is a positive-definite mass matrix. Despite its intuitive appeal and simplicity, naive SG-HMC does not preserve the target posterior [273]: stochastic-gradient noise makes the desired joint distribution $\pi(\theta, \mathbf{r}) \propto \exp(-H(\theta, \mathbf{r}))$, where $H(\theta, \mathbf{r}) = U(\theta) + \frac{1}{2} \mathbf{r}^\top \mathbf{M}^{-1} \mathbf{r}$, non-invariant. To mitigate this problem, Chen et al. [273] introduces a “friction” term $C\mathbf{M}^{-1}\mathbf{r}$. If $\hat{B}$ estimates the stochastic-gradient noise coefficient and $C \geq \hat{B}$, the practical dynamics are

$$d\theta = \mathbf{M}^{-1} \mathbf{r} dt \quad \text{and} \quad d\mathbf{r} = -\nabla_{\theta} \tilde{U}(\theta) dt - C\mathbf{M}^{-1} \mathbf{r} dt + \sqrt{2(C - \hat{B})} d\mathbf{W}_t. \quad (2.6)$$

Here, $\mathbf{W}_t$ is standard Brownian motion. The friction and injected-noise terms counteract the stochastic-gradient noise. These modified dynamics are commonly known as second-order Langevin dynamics [1707]. The complete SG-HMC framework is outlined in Algorithm 9.

Algorithm 9: Stochastic Gradient Hamiltonian Monte Carlo (SG-HMC)

Inputs : dataset $\mathcal{D}$, initial sample $\theta_0 \in \Theta$, initial momentum $\mathbf{r}_0$, step-size schedule $\{\alpha_t\}_{t \geq 0}$, inner steps m , friction coefficient C , noise estimate $\hat{B}$

Outputs: collected samples $\mathcal{S} \subset \Theta$

$\theta \leftarrow \theta_0; \mathbf{r} \leftarrow \mathbf{r}_0; \mathcal{S} \leftarrow \emptyset$

for each iteration do

$\mathbf{r} \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$

$(\theta^{(0)}, \mathbf{r}^{(0)}) \leftarrow (\theta, \mathbf{r})$

for $i = 1$ **to** m **do**

$\Xi \leftarrow$ a mini-batch sampled from $\mathcal{D}$

$\tilde{U} \leftarrow -\frac{|\mathcal{D}|}{|\Xi|} \log p(\Xi | \theta^{(i-1)}) - \log p(\theta^{(i-1)})$

$\theta^{(i)} \leftarrow \theta^{(i-1)} + \alpha_t \mathbf{M}^{-1} \mathbf{r}^{(i-1)}$

$\mathbf{r}^{(i)} \leftarrow \mathbf{r}^{(i-1)} - \alpha_t \nabla_{\theta} \tilde{U} - \alpha_t C \mathbf{M}^{-1} \mathbf{r}^{(i-1)} + \sqrt{2\alpha_t(C - \hat{B})} \epsilon_t$

// Optionally resample momentum

// Compute energy

// $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$(\theta, \mathbf{r}) \leftarrow (\theta^{(m)}, \mathbf{r}^{(m)})$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\theta\}$

// No Metropolis-Hastings step

2.3 Practical Concerns and Recent Improvements

SG-MCMC methods have been shown to be effective on many small-scale deep learning models [273, 430, 1737]. However, their application to large-scale deep learning is hindered by several practical concerns. In this section, we discuss these concerns and selected attempts to address them. Chapter 5 presents a complementary perspective on scalable, low-precision sampling.

2.3.1 Sampling from Diverse Local Minima

The loss surface (or energy landscape) of deep learning models is highly non-convex and multimodal [1029, 1854], which makes it difficult for sampling algorithms to move between modes. Moreover, a loss surface typically contains multiple local minima that may all represent good solutions. However, traditional SG-MCMC samplers commonly use a decaying step size and converge to only one local minimum, reducing their ability to explore the entire loss surface.

To mitigate this problem, Zhang et al. [1854] proposed cyclical stochastic gradient MCMC (cSG-MCMC), which uses the cyclical step-size schedule shown in Figure 2.1. The schedule is defined as

$$\alpha_t = \frac{\alpha_0}{2} \left[\cos \left(\frac{\pi \bmod(t-1, \lceil T/K \rceil)}{\lceil T/K \rceil} \right) + 1 \right], \quad (2.7)$$

where α_0 is the initial step size, K is the number of cycles, and T is the total number of iterations. The cyclical step-size schedule enables the sampler to leave the current mode when the step size increases. It can therefore explore and characterise multiple modes while retaining asymptotic convergence to the target distribution.

2.3.2 Sampling from Wide and Robust Local Minima

Distribution shift between training and test data creates a generalisation challenge for SG-MCMC algorithms [91]. Local minima identified using the training data may not remain local minima under the test data, especially when they are sharp [82, 263]. To mitigate this problem, Li and Zhang [1023] introduces Entropy-MCMC (EMCMC), which incorporates flatness-aware optimisation methods [163, 263, 544] into the SG-MCMC framework. Specifically, EMCMC introduces local entropy into the posterior distribution of model parameters:

$$p(\tilde{\theta} | \mathcal{D}) = p(\theta, \theta_a | \mathcal{D}) \propto \exp \left\{ -f(\theta) - \frac{1}{2\eta} \|\theta - \theta_a\|^2 \right\}, \quad (2.8)$$

where $f(\theta) = U(\theta)$ is the negative log-posterior energy, $\eta > 0$ is a coupling constant, and θ_a is an auxiliary parameter vector that locally explores the neighbourhood of θ . This posterior induces an extended energy whose gradient is

$$\nabla_{\tilde{\theta}} U(\tilde{\theta}) = \begin{bmatrix} \nabla_{\theta} U(\tilde{\theta}) \\ \nabla_{\theta_a} U(\tilde{\theta}) \end{bmatrix} = \begin{bmatrix} \nabla_{\theta} f(\theta) + \frac{1}{\eta}(\theta - \theta_a) \\ \frac{1}{\eta}(\theta_a - \theta) \end{bmatrix}. \quad (2.9)$$

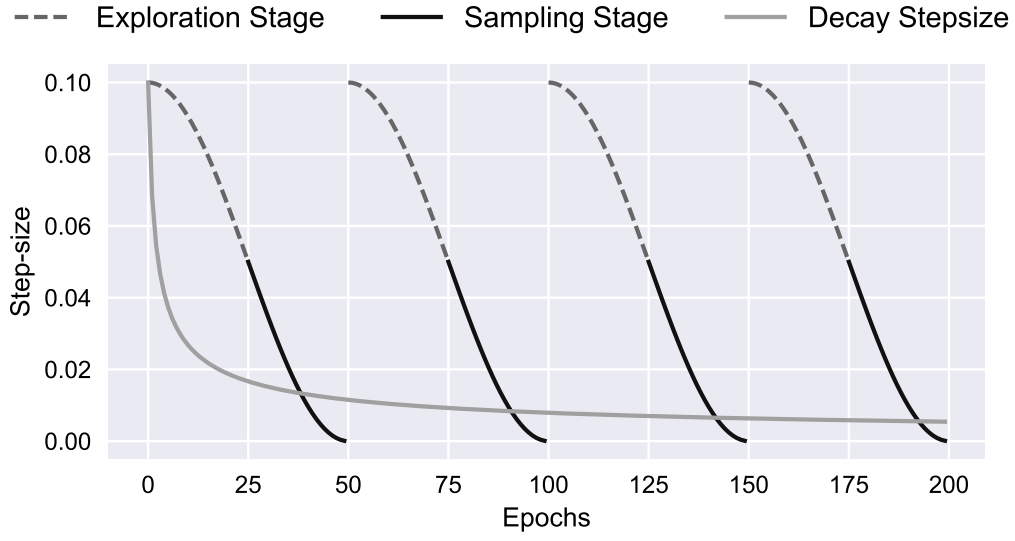


Figure 2.1: Comparison of cyclical and decaying step-size schedules. Adapted from [1854] with the permission of Ruqi Zhang.

This form provides a clear interpretation of the EMCMC sampling process: i) θ seeks low-energy regions of the energy landscape, and ii) θ_a acts as a “direction-correction force” that pulls θ away from sharp minima. Figure 2.2 illustrates this process. EMCMC frames flatness-aware sampling as a standard MCMC process on an extended energy function without introducing an additional correction step. It also has convergence guarantees and can converge faster than previous flatness-aware methods [263, 483].

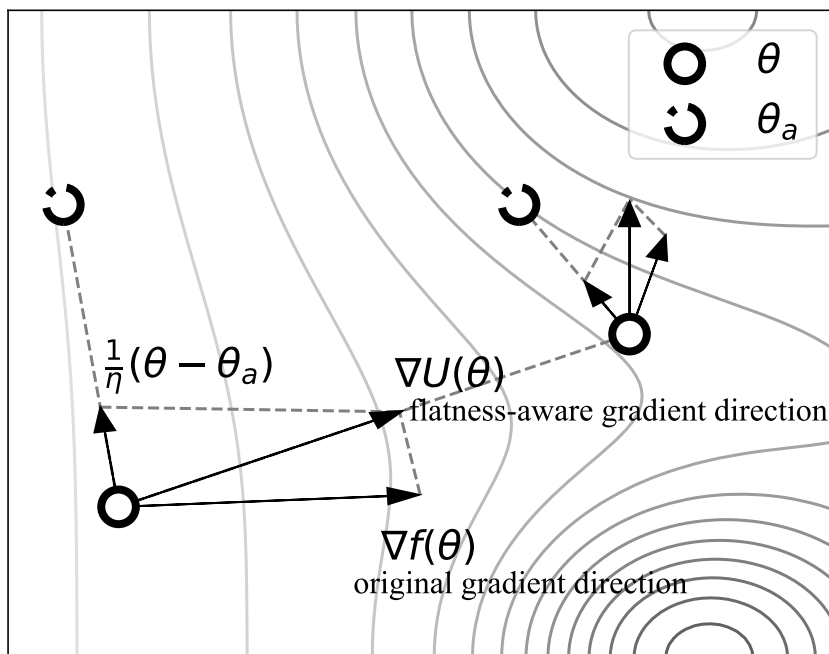


Figure 2.2: The sampling dynamics of Entropy-MCMC, showing how the guiding variable θ_a pulls θ toward flat regions in the energy landscape. Adapted from Li and Zhang [1023] with the permission of Bolian Li.

Chapter 3

Simulation-based inference

Simulation-based inference (SBI) [344, 400, 1520] has the same goal as Bayesian inference: Given observed data and a set of beliefs, apply Bayes’ rule to update those beliefs accordingly. Unlike other Bayesian inference methods such as Markov chain Monte Carlo (MCMC) or variational inference, SBI does not rely on evaluations of the likelihood, but only on samples thereof. This makes SBI applicable to any kind of black-box function that takes in parameters and simulates observational data, a so-called forward simulator.

As an illustrative example, imagine that you are an astronomer interested in the study of planets that orbit distant stars, i.e., exoplanets. Exoplanets are usually obscured by the bright light emanating from their host stars. Astronomers observe the brightness and spectrum of these stars in a time series called a light curve, and aim to infer from these light curves properties of the exoplanets using physical models. Suppose we want to estimate the mass of an exoplanet θ_o from the light curve observations x_o of a distant star. Using relevant physical theory, we design a computer program that maps a hypothetical mass θ to a simulated light curve x (the “forward model”). By running such a simulator, we can obtain samples x from the likelihood $p(x | \theta)$, but evaluating the likelihood can be computationally expensive or impossible (e.g., if the simulator takes a long time to run, or if it contains many latent variables). *Simulation-based inference* computes the posterior distribution $p(\theta | x_o)$ while only relying on samples (or simulations) from the likelihood. To achieve this, recent SBI methods use neural networks to parameterise an “inference model” $q_\psi(\theta | x_o)$, where ψ denotes the neural-network weights.

3.1 Introduction

Classical approaches for Bayesian inference (e.g., MCMC; Chapter 1, or variational inference; Chapter 12) rely on repeated evaluations of the likelihood and prior in order to characterise the posterior distribution and to generate samples from it. While powerful, these approaches face limitations for some forward models: Many models in science and engineering are defined as stochastic simulators, which can efficiently be run in the forward direction (i.e., their likelihood can be *sampled* from), but whose likelihood (as well as its gradient with respect to parameters) cannot be evaluated, or only with great computational cost [1088, 1287, 1520]. For example, the simulator might sample a long series of latent variables $z_{1:N}$ before generating observations. Evaluating the likelihood of an observation $p(x_o | \theta)$ would require integrating out these latent variables via

$$p(x_o | \theta) = \int p(x_o, z_{1:N} | \theta) dz_{1:N},$$

which can be computationally prohibitive even for a moderate number of latent variables. In addition, many classical Bayesian inference methods are expensive at test-time: Every time we want to perform inference on new data, we have to re-run the inference procedure, which might involve computationally expensive iterations (e.g., in MCMC sampling). As a result, it has been challenging to tackle high-throughput or real-time applications with Bayesian inference.

Simulation-based inference (SBI) methods overcome these limitations. The basic goal of SBI is to perform inference given only access to simulations (i.e., samples from the likelihood), without having to explicitly evaluate the likelihood, or any access to the internal workings of the forward model or its gradients. In addition, many SBI methods also *amortize* the cost of Bayesian inference: After an initial phase of simulation and training, they can perform inference for any new dataset within milliseconds, typically requiring only a single forward pass through a deep neural network at test time.

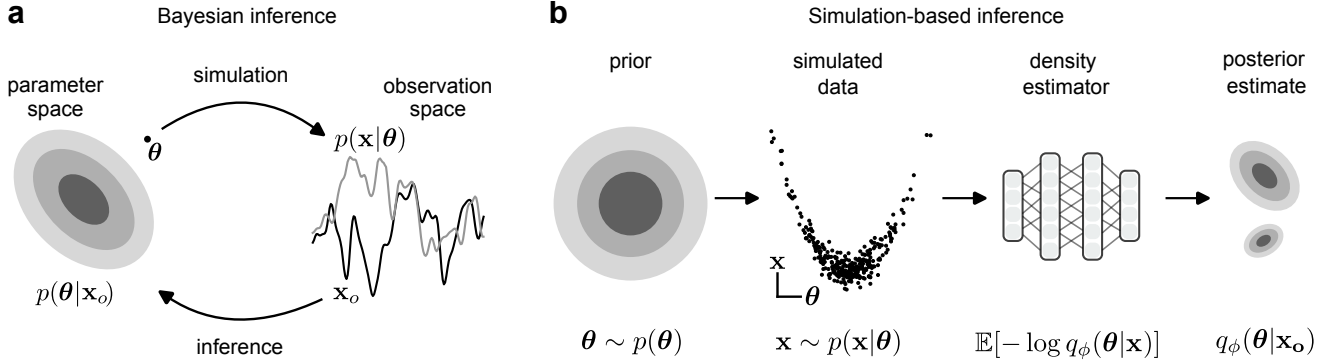


Figure 3.1: *a*: Simulators generate synthetic data x (right) given parameters θ (left). Bayesian inference computes the posterior distribution over parameters (left) given observations x_o (right). *b*: Simulation-based inference uses simulations (i.e., samples from the likelihood, but no likelihood evaluations) to perform Bayesian inference. A recent SBI method, Neural Posterior Estimation (NPE) uses neural networks to improve inference. NPE draws samples from the prior and runs the simulator to generate pairs of (θ, x) . These data are then used to train an inference network: A conditional generative model that estimates the posterior distribution given any data x . After training, the inference network is evaluated at any observation in order to draw samples from the posterior, without requiring further simulations or training. Figure modified from Deistler et al. [400].

Early SBI methods were developed for cases in which the likelihood of the model cannot be evaluated, and are often referred to as *Approximate Bayesian Computation* methods [107, 1132, 1520]. Wasserstein ABC provides an example of comparing observed and simulated empirical distributions directly, avoiding the need to choose summary statistics [131]. More recently, the ability of SBI methods to perform inference on any black-box model, to massively parallelise model runs, and to amortize inference has expanded their scope. For example, even when the likelihood or model gradients can—in principle—be evaluated, this might require interfacing with code describing the forward model written across decades in outdated programming languages. In contrast, SBI methods entirely decouple the inference process from simulation runs, which makes it easy to apply SBI methods to simulators that are written in other programming languages or that have to be run on dedicated hardware (or, in principle, which might even correspond to a physical or living system generating the simulations from input). In other scenarios, the likelihood *can* be evaluated, but one needs fast inference for real-time or high-throughput analysis of observed phenomena, which is often out of reach for, e.g., MCMC sampling, but is possible with amortized SBI methods [378]. Finally, in some cases, large-scale training can also yield highly accurate and robust inference results [378, 399, 1092]. Likely due to its simplicity, the core idea of SBI—to use forward simulations to train a supervised machine learning model—has been used in modified ways across many domains, for example, for tabular foundation models [771].

The core insight of SBI is that performing Bayesian inference can be seen as a form of performing *conditional density estimation*. In recent years, a flurry of powerful conditional density estimation methods—such as normalizing flows—have been developed, often under the moniker of “Generative AI”. This has enabled an explosion of powerful new SBI methods which build on these advances, and which have been successfully applied in a wide range of scientific disciplines, e.g., in neuroscience [325, 399, 644, 674, 712, 1426], cognitive science [177, 1686], biology [65, 920], and physics [99, 148, 324, 378, 379, 588, 700, 1012, 1188]. Here, we introduce the basic ideas and concepts of SBI. We focus on *Neural Posterior Estimation*, a conceptually simple yet powerful algorithm, but also review other approaches, diagnostic tools, and current developments. Two notable methods are *Neural Likelihood Estimation* [1291] and *Neural Ratio Estimation* [471, 741, 1175, 1601], which we explain later in the chapter.

3.2 Simulation-based inference with Neural Posterior Estimation

3.2.1 Neural Posterior Estimation

The goal of Bayesian inference is to calculate—at least approximately—the posterior distribution of the form $p(\theta | x)$. Via Bayes’ rule, the posterior is proportional to the product of the prior and the likelihood $p(\theta | x) = \frac{1}{Z(x)} p(x | \theta) p(\theta)$, where $Z(x) = \int p(x, \theta) d\theta = p(x)$ is the normalizing constant, also called the marginal likelihood or evidence. Calculating $Z(x)$ is often challenging, and many classical Bayesian inference algorithms such as MCMC sampling therefore aim to obtain samples from the posterior without having to evaluate it. However, even

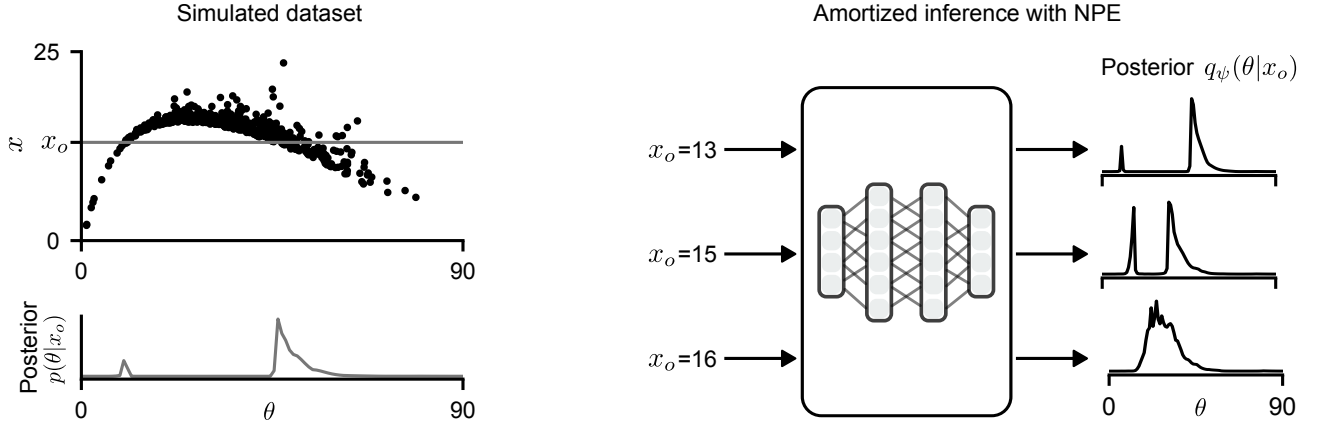


Figure 3.2: Framing Bayesian inference as a supervised learning problem. Left: A database of simulations (θ, x) , sampled from the joint distribution $p(\theta, x)$ by drawing samples from the prior and running the simulator. The posterior distribution (left, bottom) is a horizontal cut through the joint distribution. Right: NPE trains an inference network on these simulations. After training, the network can be evaluated at any observation x_o and directly predicts the posterior distribution. Figure modified from Deistler et al. [400].

these algorithms require (typically many) evaluations of the likelihood $p(x | \theta)$. However, many models in scientific applications are defined through numerical simulators: One can easily generate *samples* from the likelihood $p(x | \theta)$ by running the simulator, but *evaluating* the likelihood can be expensive or even impossible. How can one perform Bayesian inference in such setting?

A key insight is that one does not need Bayes rule for Bayesian inference: The posterior distribution $p(\theta | x_o)$ at some particular $x = x_o$ is, by definition, a *conditional distribution*, i.e., a normalised slice through the joint distribution $p(\theta, x)$ at $x = x_o$ (Fig. 3.2, left). Thus, an alternative way to get the posterior is to generate a simulated dataset with—possibly many—paired samples $D = \{(\theta_i, x_i)\}_{i=1}^N$, and then to directly estimate this posterior distribution via conditional density estimation. We can generate such a simulated dataset by first sampling the θ_i from the prior, and then, for each θ_i , to run the simulator once to get the associated x_i , as $p(\theta, x) = p(\theta)p(x | \theta)$.

We want to learn a mapping from x to the distribution $p(\theta | x)$. One way to tackle this is to parameterise the posterior distribution by a functional form, for example, a Gaussian with mean $\mu(x)$ and covariance $\Sigma(x)$, and to use a neural network (with parameters ψ) to capture the mapping from x to associated posterior parameters $(\mu_\psi(x), \Sigma_\psi(x))$. One can learn ψ by maximising the log-likelihood of the parameters, i.e., by minimising the loss

$$\mathcal{L}(\psi) = -\frac{1}{N} \sum_{i=1}^N \log q_\psi(\theta_i | x_i), \quad (3.1)$$

where in this case, the predicted posterior probabilities $q_\psi(\theta | x)$ would be given by a Gaussian, $q_\psi(\theta | x) = \mathcal{N}(\theta | \mu_\psi(x), \Sigma_\psi(x))$ [172, 1288]. Thus—and provided that one can generate a large set of simulations D —we have turned Bayesian inference into a supervised learning problem.

Clearly, a Gaussian might be a poor approximation to the posterior, but this approach can readily be generalised to more flexible models. For example, one can use a Mixture of Gaussians [159, 1288], which requires the network to learn the mapping to multiple means $\mu_j(x)$, covariances $\Sigma_j(x)$ and associated mixture weights $\pi_j(x)$. In recent years, more flexible and easy-to-train neural conditional density estimators have been proposed, which can be used for estimating non-Gaussian posteriors. In particular, *Normalizing Flows* [1290] constitute a class of neural conditional densities estimators which have many favourable properties for SBI, as we will describe in the next section.

Once the conditional density estimator (sometimes referred to as *inference network*) has been trained on simulated data D , it can readily be *evaluated* on any new observation x_o to approximate the posterior $p(\theta | x_o) \approx q_\psi(\theta | x_o)$. Indeed, it can be shown that—provided that the density estimator is sufficiently flexible, that the model is well specified, and the optimisation identifies the minimum—it will yield the true posterior [1288]. To demonstrate this, we rewrite the loss function (in the limit of infinite training data, i.e., in expectation of samples

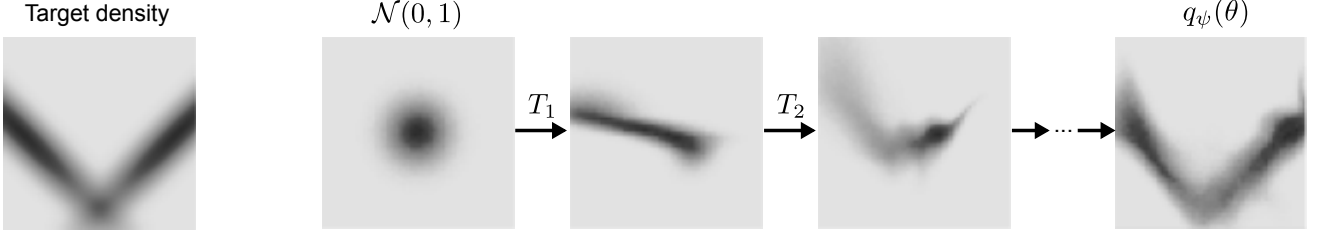


Figure 3.3: Normalizing flows. Left: A target distribution which we aim to estimate based on samples. Right: Normalizing flows define a base distribution (e.g., a multivariate Gaussian) and transform this distribution through multiple transformations T_i . After training, the transformed distribution $q_\psi(\theta)$ approximates the target distribution.

from $p(\theta, x) = p(\theta)p(x | \theta)$ as

$$\begin{aligned}
 \mathcal{L} &= \mathbb{E}_{p(\theta)p(x|\theta)}[-\log q_\psi(\theta | x)] \\
 &= \iint -p(\theta)p(x | \theta) \log q_\psi(\theta | x) \, d\theta \, dx \\
 &= \iint -p(\theta) \frac{p(\theta | x)p(x)}{p(\theta)} \log q_\psi(\theta | x) \, d\theta \, dx \\
 &= \int p(x) \int -p(\theta | x) \log q_\psi(\theta | x) \, d\theta \, dx.
 \end{aligned}$$

The inner integral differs from the Kullback–Leibler divergence $D_{\text{KL}}(p(\theta | x) \| q_\psi(\theta | x))$ only by the conditional entropy of $p(\theta | x)$, which does not depend on the neural-network parameters ψ . Therefore, if the conditional density estimator is well specified, the population loss is minimised when $q_\psi(\theta | x) = p(\theta | x)$ for $p(x)$ -almost every x .

One big advantage of these approaches is that—once the inference network has been trained—the posterior distribution for new observations x_o can be evaluated by just plugging it into the inference network to get $q_\psi(\theta | x)$. Thus, inference can be performed with a single forward pass through a neural network, i.e., it is *amortized* (Fig. 3.2, right). Conversely, a drawback of this approach is that our simulations need to cover the entire prior, which might be wasteful as we might generate many simulations which look very different to an observation x_o we are interested in, and which therefore do not help in constraining the relevant posterior. To address this issue (but which requires giving up on amortization), so-called *sequential* approaches have been developed, in which simulations are generated adaptively to target a posterior for a specific observation. We discuss these approaches in Section 3.3.

The approach described above is now commonly known as *Neural Posterior Estimation* (NPE). Given its conceptual simplicity and the many variations that it allows, it is challenging to track down its precise origins, and variants of it have likely been discovered independently in different application domains. Beaumont et al. [107] and Blum and François [172] introduced the idea of fitting a regression model from data to parameters to refine rejection-ABC approaches, which can be interpreted as NPE with a Gaussian inference network. Papamakarios and Murray [1288] showed that flexible density estimators can directly learn the posterior, when optimised with log-loss minimisation (also see Le et al. [991]), and introduced a sequential version. Lueckmann et al. [1090] (which, to our knowledge, introduced the term (S)NPE) introduced an alternative sequential formulation, and also embedding networks for learning summary statistics from time series data. Greenberg et al. [663] provided the first method that could use normalizing flows and is compatible with sequential learning, and used convolutional embedding nets. Ardizzone et al. [45] proposed using invertible neural networks for solving inverse problems, resulting in an algorithm very similar to flow-based NPE, and Radev et al. [1359] popularised the amortization property of NPE.

3.2.2 Conditional density estimation with normalizing flows

As described above, a core task in SBI is to estimate the conditional distribution of parameters given some observations. Normalizing flows [1290] have emerged as a popular conditional density estimator for SBI. They are simple and efficient to train, can be used to quickly generate samples, or to evaluate exact log-probabilities of the learned density. The latter can be useful for many downstream analyses such as maximum-a-posteriori estimation or visualisation of the posterior distribution. Finally, normalizing flows are (typically) trained by minimising the negative log-likelihood as loss function. This loss function is mass-covering, which, as we will discuss below, can

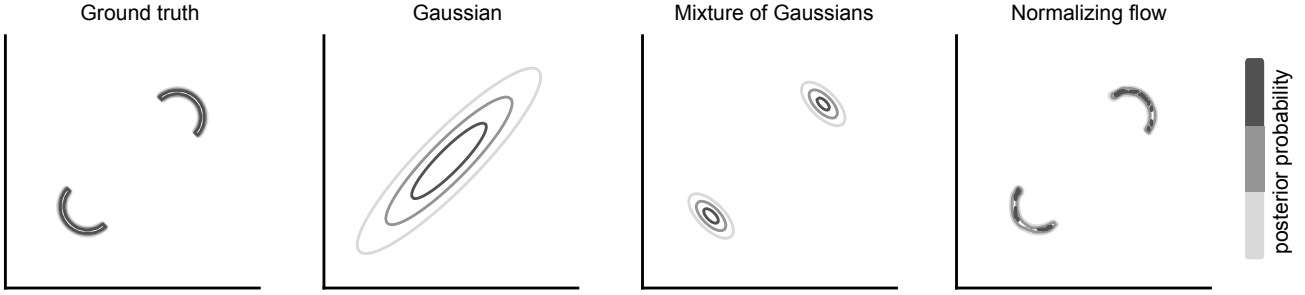


Figure 3.4: Expressiveness of different generative models for approximating the posterior distribution with NPE. The ground truth posterior (left) consists of two moons and has low posterior density in between. Using a Gaussian (second from left) as generative model produces a poor—albeit mass-covering—posterior distribution. A mixture of multiple Gaussians (second from right) improves accuracy, but fails to produce the moon shapes. Normalizing flows (right) enable NPE to accurately capture complex posterior distributions.

be beneficial for SBI.

Normalizing flows model a target density $q(\theta)$ by learning a transformation $T : Z \rightarrow \Theta$ such that a random variable Z , which is defined to follow a simple “base” distribution $p(z)$ (typically a multivariate standard Normal distribution) is transformed into a random variable Θ , such that the probability density of Θ is $q(\theta)$. The probability density modelled by such a transformation can be computed with the change of variables formula

$$q(\theta) = \left| \det \left(\frac{\partial T^{-1}}{\partial \theta} \right) \right| p_Z(T^{-1}(\theta)), \quad (3.2)$$

where T^{-1} is the inverse of the transformation T and $\det \left(\frac{\partial T^{-1}}{\partial \theta} \right)$ is the determinant of the Jacobian of the inverse transformation T^{-1} . Normalizing flows stack many transformations T in order to model complicated densities.

To evaluate Eq. 3.2 and to efficiently train normalizing flows, the transformation T must be a diffeomorphism (i.e., it must be differentiable and invertible). In addition, the Jacobian of T^{-1} is of size $N \times N$ (for densities of dimensionality N), and computing the determinant of such a matrix is typically of computational complexity $\mathcal{O}(N^3)$, which would be prohibitive for large N . As such, normalizing flows require that the transformation is constructed such that the determinant of the Jacobian can be evaluated efficiently. Normalizing flows differ in how they define the transformation T such that it fulfills these criteria. A popular class of transformations are autoregressive flows, and within SBI, masked autoregressive flows [1289] and neural spline flows [470] are particularly popular. These normalizing flows define bijective transformations for every individual dimension and then couple dimensions with neural networks.

Normalizing flows train the transformations T by minimising the negative log-likelihood of the data

$$\psi = \arg \min_{\psi} - \log \prod_{i=1}^M q_{\psi}(\theta_i).$$

In neural posterior estimation (NPE), normalizing flows are used to estimate the posterior probability density $p(\theta | x)$. To estimate a *conditional* probability density, NPE conditions that transformation T on data x , or an embedding $s(x)$ thereof (Sec. 3.2.3). This enables NPE to learn different posterior distributions for different data.

The normalizing flow is then trained over pairs of parameters and simulation outputs, where the parameters are sampled from the prior. NPE minimises the loss

$$\mathcal{L} = \mathbb{E}_{p(\theta)p(x|\theta)} [-\log q_{\psi}(\theta | x)]. \quad (3.3)$$

If the normalizing flow is well-specified, i.e., the true conditional posterior $p(\theta | x)$ belongs to the family of normalizing-flow models, then there exists some $\psi^* \in \Psi$ such that $q_{\psi^*}(\theta | x) = p(\theta | x)$ for $p(x)$ -almost every x , where Ψ denotes the parameter space. In this case, the population loss is minimised at the true conditional posterior, assuming successful optimisation. After training, one can then sample from the posterior for any observation x_o by drawing a sample from the base distribution and transforming it with the series of transformations T (which are conditioned on x_o). Notably, samples from normalizing flows are i.i.d. and follow the estimated density exactly (i.e., normalizing flows do not require any approximations to draw samples from the modelled density). In addition, normalizing flows can also evaluate the density of parameters under the posterior via Eq. 3.2.

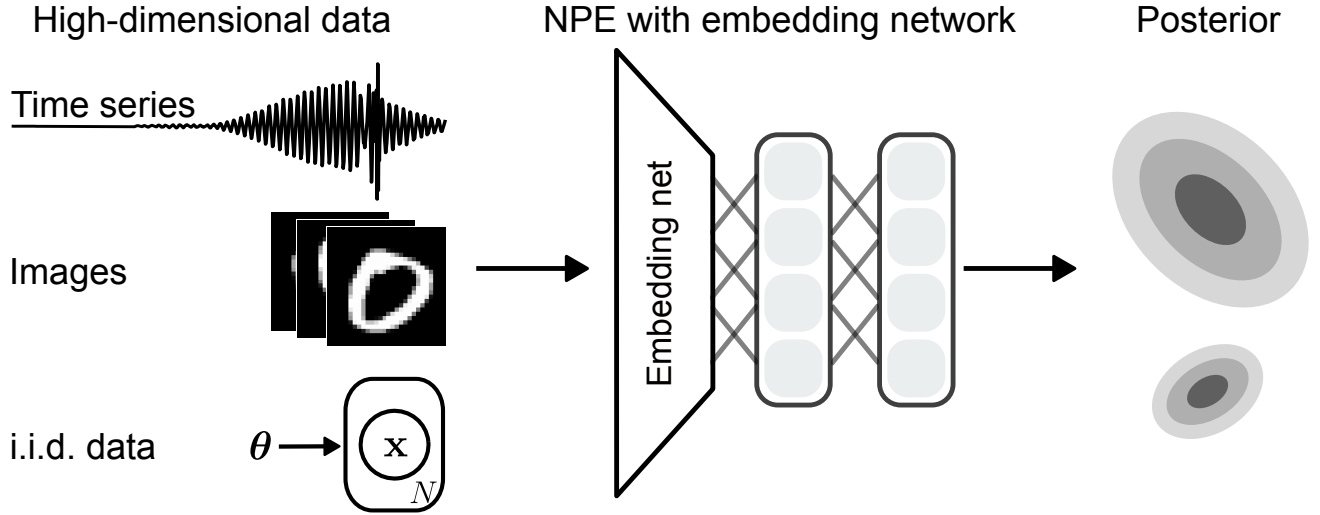


Figure 3.5: Neural Posterior Estimation (NPE) with embedding networks. Left: For many models, simulation outputs can be high-dimensional. For example, simulators might produce long time series, images, or i.i.d. data given a particular parameter set. Middle: In order to perform inference based on such observations, NPE can be combined with suitable embedding networks that efficiently reduce these data to lower-dimensional summary statistics. These summary statistics are then processed by the inference network, and both neural networks can be trained end-to-end. Right: Illustration of a posterior distribution underlying such observations.

An important property of this loss is that it tends to be mass-covering: it will strongly penalise the normalizing flow being too narrow (as $\log q(\theta | x)$ becomes a large negative number, leading to very high loss). As such, for limited training data or for imperfect convergence of the normalizing flow, $q(\theta | x)$ will tend towards being broader than the true posterior distribution $p(\theta | x)$. In many applications of SBI, this is a desirable feature: too narrow posteriors would indicate overly confident parameter estimates, which could lead to wrong claims of scientific discovery [742]. This mass-covering behaviour is in contrast to variational inference, which typically uses divergence objectives whose covering or seeking behaviour depends on the divergence direction, with common formulations tending to be mode-seeking (i.e., towards a too narrow posterior approximation) [1035]. Since we operate with finite samples, none of these are guarantees.

3.2.3 Learning from high-dimensional data with embedding networks

Many models in science and engineering produce high-dimensional simulation outputs such as images or long time series. In order to estimate parameters underlying such data, NPE can be combined with *embedding networks* (Fig. 3.5) [1090]. Embedding networks are neural networks which take as input high-dimensional data x and return a lower-dimensional summary statistic $s(x)$ of them. These summary statistics are then passed to the inference network, which uses the summary statistics to approximate the posterior. Ideally, these summary statistics will be (approximately) sufficient, i.e., still preserve all relevant information for the posterior, so that $p(\theta | x) \approx p(\theta | s(x))$ [280, 508].

In NPE, the embedding network can be trained end-to-end with the inference network, that is, with the log-likelihood loss specified in Eq. 3.3. Provided that the embedding network is sufficiently expressive, it will automatically learn appropriate summary statistics, without requiring separate data-compression methods [280] (although those might still be empirically useful in some cases [281]). The choice of embedding network depends on the type of data modelled by the simulator. For example, for images, a popular choice for the embedding net are convolutional neural networks (CNNs), and recurrent neural networks (RNNs) might be useful for time-series data [1090].

An important case is the setting in which the observation x_o does not constitute a single observation, but rather stands for a *set* of observations $x = \{x_1, \dots, x_N\}$ which are thought to be sampled i.i.d. from the same parameter. In this case, the posterior distribution should be invariant to the ordering of the observations, which can be achieved by using a permutation invariant embedding network such as a set transformer [1003], as proposed by Chan et al. [255] for applications in population genetics. Radev et al. [1359] proposed and evaluated using such networks for datasets with varying size.

Instead of learning summary statistics end-to-end with an embedding network, one can also manually extract summary statistics from the data and perform inference based on these statistics. This requires domain knowledge and, if the choice of summary statistics is poor, might lead to a significant loss of information about parameters, but it can be beneficial if one is interested in inferring parameters given particular features of data. Indeed, in many cases, the simulator is not able to reproduce *all* features of the (experimentally) observed data. Manually defined summary statistics can avoid such misspecification by focusing only on properties of the data which can be modelled by the simulator.

3.3 Amortized and sequential inference

As described above, NPE *amortizes* the cost of inference: After training on a simulated dataset generated from prior predictives (i.e., simulation results based on parameters drawn from the prior distribution), they can perform inference for *any* new observed data x_o . In cases where one is interested in performing inference for many different observations, or where inference is time-critical, this can be highly beneficial. In cases where one only aims to perform inference for a single observation (or for few observations), NPE can be wasteful: By training a neural network on *prior* predictives, the neural network has to learn from a broad range of (simulated) data, even though it is eventually only evaluated at few observations.

Sequential methods have been developed to improve the simulation-efficiency of SBI in these cases. These methods draw parameters (which are then used to generate the training dataset) from a “proposal” distribution. This proposal distribution is chosen such that simulation outputs are expected to be closer to the observation than prior predictives. A popular choice for the proposal distribution is the posterior distribution obtained by running NPE with a limited number of simulations (but explicit active learning schemes have also been proposed [668]).

These methods are often called *sequential*, as the inference network is trained across multiple, sequentially simulated datasets, where the approximate posterior after each round guides generation of the next dataset. It has been demonstrated that sequential methods can improve simulation efficiency, often by an order of magnitude or more [398, 635, 1092].

While potentially reducing the number of required simulations, using sequential methods with NPE can have a drawback: Drawing parameters from a proposal distribution (instead of from the prior) biases the posterior distribution towards regions that were oversampled in the proposal (compared to the prior). Several methods have been developed to overcome this. Papamakarios and Murray [1288] proposed to train the neural network with the standard log-likelihood loss, and to then correct the posterior post-hoc. Lueckmann et al. [1090] suggested to importance-weight the loss, such that the density estimator directly approximates the posterior after training. Greenberg et al. [663] introduced a contrastive loss, which implicitly frames conditional density estimation as a classification problem [471], and which can be combined with arbitrary acquisition functions [668]. Finally, Blum and François [172] and Deistler et al. [398] proposed to draw parameter sets from a restricted region of the prior, such that no modifications of the loss function or post-hoc corrections are required. Notably, for many other SBI methods, such as Neural Likelihood Estimation, NLE [1091, 1291], or Neural Ratio Estimation, NRE [471, 741, 1175, 1601], parameters for the training dataset can be drawn from *any* distribution and the neural networks can nonetheless be trained with standard loss functions, as the prior is explicitly taken into account during inference with MCMC or variational inference.

3.4 Evaluating the correctness of SBI methods

3.4.1 The problem of evaluating SBI methods

After having obtained an approximate posterior with SBI methods, a central challenge is to evaluate the quality of this posterior. This is difficult because, for any real-world task, the ground-truth posterior is unavailable.

Several diagnostic tools have been developed to overcome this limitation. Note, however, that many of these tools do not provide *sufficient* conditions for posterior correctness. Instead, they provide *necessary* conditions for posterior correctness, and thus enable detection of inaccuracies in the posterior estimate. Other methods can, in principle, provide sufficient conditions for correctness, but rely on additional hyperparameters or the accuracy of other trained neural networks, which can compromise the accuracy of the diagnostic tool. Below, we focus on a set of diagnostic tools called coverage diagnostics, and we then briefly describe other methods.

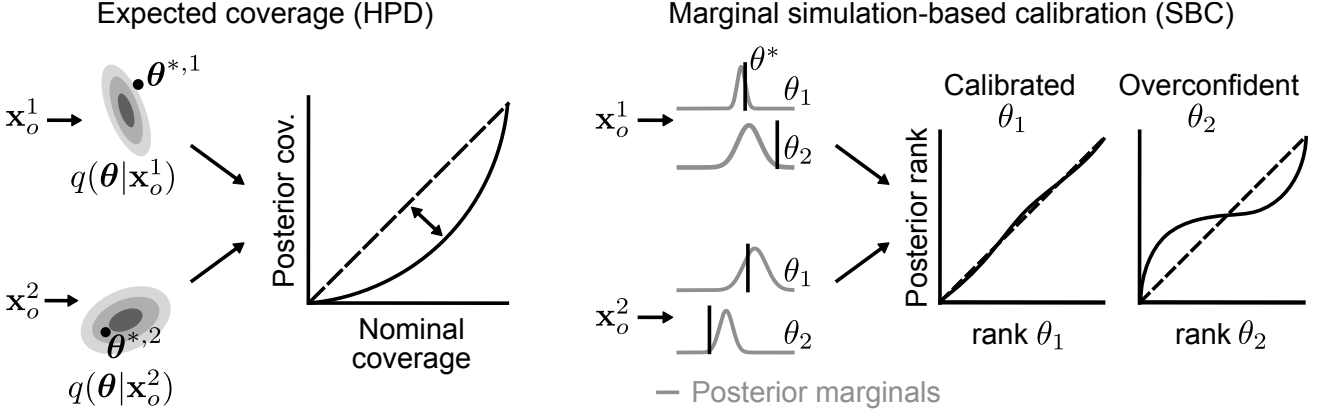


Figure 3.6: Coverage diagnostics for identifying inaccuracies in the approximate posterior. Left: Expected coverage based on the highest-probability density (HPD) region aims to detect whether the joint posterior distribution is over- or under-confident. Marginal simulation-based calibration (SBC) aims to detect inaccuracies in the marginals of the posterior distribution. Both of these methods produce rank distributions, which are optimal if the posterior lies on the diagonal, and indicate issues if it lies above or below the diagonal. Figure modified from Deistler et al. [400].

3.4.2 Coverage diagnostics

A set of diagnostic tools that aim to detect inconsistencies in the posterior distribution are *coverage diagnostics* [328, 1585]. These methods begin by generating a calibration dataset: They sample parameters from the prior and run the simulator to generate samples $\theta, x \sim p(\theta, x)$. For every x in the calibration dataset, these methods then draw samples from the posterior $\theta_{\text{post}} \sim q_\psi(\theta | x)$ obtained with an SBI method. For amortized SBI methods, this process is fast, as inference can be performed for any x without retraining or resimulating.

Coverage diagnostics then reduce posterior samples θ_{post} and the ground truth parameters θ to 1-dimensional quantities, via a reducing function $f(\cdot) : \mathbb{R}^N \rightarrow \mathbb{R}$, where N is the parameter dimensionality. The posterior approximation is *calibrated* if, for any reducing function $f(\cdot)$, the rank of $f(\theta)$ is distributed uniformly within $f(\theta_{\text{post}})$ (Fig. 3.6). Different diagnostic tools use different mappings $f(\cdot)$ to detect posterior inconsistencies. For example, $f(\theta)$ can compute the posterior log-probability $q_\psi(\theta | x)$ (often referred to as *expected coverage* based on the highest-probability density (HPD) region) [398, 742, 1174], or it can pick the 1D marginals of θ [328, 1585]. In the former case, coverage diagnostics can detect inconsistencies in higher-order moments of the posterior (Fig. 3.6, left). In the latter case, coverage diagnostics cannot detect inconsistencies in higher-order moments of the posterior, but they can provide intuition about which parameter is inconsistent (Fig. 3.6, right).

A core advantage of coverage diagnostics is that they can be run with relatively small calibration sets (typically, ~ 200 simulations), and that they do not require further tuning of hyperparameters or training of a “diagnostic neural network”. This makes them robust and easy to deploy. However, coverage diagnostics typically provide a necessary, but not a sufficient condition for posterior correctness. In particular, if the approximate posterior $q_\psi(\theta | x)$ always returns the prior distribution $p(\theta)$ (instead of an approximation to the posterior), then it passes the above described coverage diagnostics. This highlights the need for additional diagnostic tools.

3.4.3 Other methods

A simple tool for diagnosing the posterior approximation $q_\psi(\theta | x)$ is to perform *Posterior Predictive Checks* (PPCs). Given an observation, these methods draw samples from $q_\psi(\theta | x_o)$, simulate them, and then compare the simulation outputs (the posterior predictives) to the observation. For every posterior sample θ , it should be possible to obtain a simulation result that closely matches the observation. If posterior predictive checks fail (e.g., because all posterior predictives are far away from the observation, or if some parameter sets can *never* match the observation), then this indicates issues in the accuracy of $q_\psi(\theta | x)$. In particular it might be trained on too few simulations, or it might hint towards *model misspecification* (a case where the observation x_o cannot be matched by *any* parameter set, as discussed in Sec. 3.6).

Beyond coverage checks and PPCs, many other diagnostic tools for SBI have been developed over the past years. For example, some local calibration diagnostics aim to provide sufficient and necessary conditions for posterior correctness [1052, 1448]. However, these methods often require to train an additional neural network and hinge

on the accuracy of that additional trained neural network. Because of this, these diagnostic tools require larger calibration datasets and might themselves have issues (e.g., due to poor convergence of the additional trained neural network).

3.5 Alternative methods

Neural Posterior Estimation (NPE) directly estimates the posterior distribution. In addition to NPE, researchers have developed other methods for SBI. These approaches train neural networks to emulate either the likelihood $p(x \mid \theta)$ (Neural Likelihood Estimation, NLE) or the likelihood-to-evidence ratio $p(x \mid \theta)/p(x)$ (Neural Ratio Estimation, NRE) and then use traditional Bayesian inference methods (e.g., MCMC) to draw samples from the posterior. Finally, some recent methods also go beyond estimating only the posterior or the likelihood(-ratio). Below, we describe these alternative methods for SBI.

3.5.1 Likelihood estimation

Neural Likelihood Estimation (NLE) uses neural density estimators (such as conditional normalizing flows) to estimate the likelihood $p(x \mid \theta)$ [1291]. This estimate can be learned from samples of the joint distribution $\{(\theta_i, x_i)\}_{i=1}^N \sim p(\theta, x)$ by optimising the negative log likelihood of simulation outputs given parameters. NLE trains a conditional density estimator $\ell_\psi(x \mid \theta)$, such as a normalizing flow, by optimising ψ towards the objective

$$\psi^* \in \arg \min_{\psi} \left[-\frac{1}{N} \sum_{i=1}^N \log \ell_\psi(x_i \mid \theta_i) \right]. \quad (3.4)$$

After training, the trained model can be used to *emulate* the simulation process by drawing samples from the likelihood approximation $\ell_\psi(x \mid \theta)$.

For posterior inference, the trained model is multiplied by the prior to obtain the (unnormalised) posterior estimate

$$\hat{p}(\theta \mid x) := \frac{1}{Z_\psi(x)} \ell_\psi(x \mid \theta) p(\theta), \quad Z_\psi(x) := \int \ell_\psi(x \mid \theta) p(\theta) d\theta. \quad (3.5)$$

In order to draw samples from the (approximate) posterior distribution $\hat{p}(\theta \mid x_o)$, a method to draw samples from an unnormalised distribution is necessary, for example, rejection sampling or Markov chain Monte Carlo (Sec. 3.5.3). Normalizing flows enable (differentiable) evaluation of the likelihood approximation $\ell_\psi(x_o \mid \theta)$, making this possible, albeit being potentially computationally expensive. Variational inference has been shown to speed up high-dimensional sampling from NLE [635, 1774]. However, any of these sampling methods are typically slow and can introduce additional errors into the inference procedure. An additional disadvantage of NLE (compared to NPE) is that it cannot be combined with an embedding network that is trained end-to-end. One can embed x_o *before* training with a separate neural network, e.g., an autoencoder, then perform NLE on the embedded simulation outputs [280].

NLE can also be used in a sequential scheme to reduce the number of required simulations for inference given a particular observation. These sequential NLE methods focus on a specific observation x_o , generating data nearby x_o iteratively [1291]. A major advantage of NLE in that scenario is that it does not require modifications of the loss function: The neural density estimator will converge to the likelihood for any distribution $\tilde{p}(\theta)$ from which parameters are drawn.

3.5.2 Likelihood-to-evidence ratio estimation

The likelihood-to-evidence ratio $r(x \mid \theta) := \frac{p(x \mid \theta)}{p(x)} = \frac{p(\theta, x)}{p(\theta)p(x)}$ is a natural estimation target for simulation-based inference because Bayes' rule gives $p(\theta \mid x) = r(x \mid \theta)p(\theta)$. Ratio estimators allow more flexible architectures than diffeomorphic normalizing flow layers with tractable determinants. However, density ratios can span many orders of magnitude and therefore be numerically difficult to approximate directly. For this reason, one typically estimates $\log r(x \mid \theta)$. A stable approach to estimating $\log r(x \mid \theta)$ converts the problem into a classification task. Several variants have been developed [343, 471, 1175, 1601]; here, we focus on the method of Hermans et al. [741].

NRE introduces a binary class label y and defines the joint distribution $\pi(\theta, x, y) := \pi(\theta, x \mid y)\pi(y)$. We assign

equal class probabilities, $\pi(y = 0) := \pi(y = 1) := \frac{1}{2}$, and define the class-conditional distributions as

$$\pi(\theta, x | y) := \begin{cases} p(\theta)p(x), & y = 0, \\ p(\theta, x), & y = 1. \end{cases} \quad (3.6)$$

$p(\theta)p(x)$ is called the product of marginals. It is best understood algorithmically: One takes two independent samples $\theta_a, \theta_b \sim p(\theta)$ and simulates one of them $x_b \sim p(x | \theta_b)$. Then the samples are distributed $(\theta_a, x_b) \sim p(\theta)p(x)$. Sampling from the joint $p(\theta, x)$ is achieved by first sampling the prior $\theta_c \sim p(\theta)$ and simulating $x_c \sim p(x | \theta_c)$. We then have a sample $(\theta_c, x_c) \sim p(\theta, x)$.

Bayes' rule shows why this classification problem identifies the desired ratio:

$$\frac{\pi(y = 1 | \theta, x)}{\pi(y = 0 | \theta, x)} = \frac{\pi(\theta, x | y = 1)\pi(y = 1)}{\pi(\theta, x | y = 0)\pi(y = 0)} = \frac{p(\theta, x)}{p(\theta)p(x)} = r(x | \theta).$$

Consequently, $\pi(y = 1 | \theta, x) = \frac{r(x|\theta)}{1+r(x|\theta)} = \sigma(\log r(x | \theta))$, where σ denotes the sigmoid function. We therefore construct

$$\hat{\pi}(y = 1 | \theta, x) := \sigma(f_\psi(\theta, x)) \quad \text{and} \quad \hat{r}(x | \theta) := \exp(f_\psi(\theta, x)),$$

where $\hat{\pi}(y = 1 | \theta, x)$ is a classifier, with $\hat{\pi}(y = 0 | \theta, x) = 1 - \hat{\pi}(y = 1 | \theta, x)$, and $\hat{r}(x | \theta)$ is a likelihood-to-evidence ratio estimate. Both are parameterised by the neural network $f_\psi(\theta, x)$ with weights ψ . We optimise ψ by minimising the conditional KL divergence,

$$\psi^* \in \arg \min_{\psi} D_{\text{KL}}(\pi(y | \theta, x) || \hat{\pi}(y | \theta, x)), \quad (3.7)$$

$$D_{\text{KL}}(\pi(y | \theta, x) || \hat{\pi}(y | \theta, x)) = \iint \pi(\theta, x) \sum_{y \in \{0,1\}} \pi(y | \theta, x) \log \frac{\pi(y | \theta, x)}{\hat{\pi}(y | \theta, x)} dx d\theta. \quad (3.8)$$

Here, $\pi(\theta, x)$ is the marginal mixture induced by the constructed classification problem:

$$\pi(\theta, x) := \pi(y = 1)\pi(\theta, x | y = 1) + \pi(y = 0)\pi(\theta, x | y = 0) = \frac{p(\theta, x) + p(\theta)p(x)}{2}. \quad (3.9)$$

The conditional KL divergence differs from the binary cross-entropy used to train the classifier only by the conditional entropy of $\pi(y | \theta, x)$, which is constant with respect to ψ . The two objectives therefore have the same minimiser. Assuming sufficient model capacity and successful optimisation, the conditional KL divergence is zero at the optimum, so $\pi(y = 1 | \theta, x) = \sigma(f_{\psi^*}(\theta, x))$ and $f_{\psi^*}(\theta, x) = \log r(x | \theta)$ almost everywhere under $\pi(\theta, x)$.

We can now form the unnormalised posterior approximation $\hat{p}'(\theta | x) := \hat{r}(x | \theta)p(\theta) = p(\theta)\exp(f_\psi(\theta, x))$. Unless the classifier recovers the exact ratio, $\hat{p}'$ need not integrate to one. Its normalised counterpart is

$$\hat{p}(\theta | x) := \frac{\hat{p}'(\theta | x)}{Z_\psi(x)} = \frac{\hat{r}(x | \theta)}{Z_\psi(x)}p(\theta) = \frac{\exp(f_\psi(\theta, x))}{Z_\psi(x)}p(\theta), \quad (3.10)$$

with $Z_\psi(x) := \int \exp(f_\psi(\theta, x))p(\theta) d\theta$. At the exact optimum, $\exp(f_{\psi^*}(\theta, x)) = r(x | \theta)$ and hence $Z_{\psi^*}(x) = 1$. In practice, posterior behaviour is usually characterised using samples, and samplers for unnormalised distributions do not require $Z_\psi(x)$ to be computed.

One can draw samples from $\hat{p}(\theta | x)$ using any sampling method for unnormalised distributions, just like with NLE (Sec. 3.5.3). The ratio-based neural network $f_\psi(\theta, x)$ can be both more expressive and cheaper than the normalizing flow-based likelihood approximation from NLE. Flows require parameterising an efficient diffeomorphism with a tractable log determinant, whereas NRE permits arbitrary neural-network architectures. One limitation is that the standard NRE objective in (3.7) does not generally attain the same asymptotic efficiency as maximum-likelihood density estimation [295, 1392, 1824]. Alternative objectives have therefore been explored for ratio estimation [634, 1177].

3.5.3 Sampling from the approximate posterior using unnormalised inference models

While NPE approximates the posterior directly, NLE and NRE require an additional inference step to draw samples from the posterior. To this end, one can use general-purpose sampling methods (see Chapter 1) and adapt them

to sample from an unnormalised posterior estimate. Rejection sampling or reweighted sampling from the prior are natural choices for simple problems with low dimensionality due to straightforward embarrassingly parallel implementations [1211]. Both of these methods start by drawing N samples from the prior

$$\theta_i \sim p(\theta), \quad i \in \{1, 2, \dots, N\}. \quad (3.11)$$

In *rejection sampling*, we choose a constant $M \geq 1$ such that $M \cdot p(\theta) \geq \hat{p}'(\theta | x_o)$ for all θ in the support of $\hat{p}'(\theta | x_o)$, where $\hat{p}'(\theta | x_o)$ denotes the unnormalised posterior density. For each of the N samples, we draw

$$u_i \sim \mathcal{U}(0, 1), \quad i \in \{1, 2, \dots, N\}, \quad (3.12)$$

where we let $\mathcal{U}(a, b)$ denote the one-dimensional uniform distribution between a and b . This algorithm yields samples from the approximate posterior $\hat{p}(\theta | x_o)$ in the following way:

$$u_i < \frac{\hat{p}'(\theta_i | x_o)}{M \cdot p(\theta_i)} \implies \theta_i \sim \hat{p}(\theta | x_o). \quad (3.13)$$

Rejection sampling generates N/M samples, on average. It is efficient when $M \approx 1$, *i.e.*, in low dimensions and when the prior and approximate posterior have similar support. Rejection sampling is inefficient when the approximate posterior is narrow compared to the prior.

In *reweighted sampling*, we use the samples from the prior in (3.11), but assign them a self-normalised importance weight. The weights and self-normalised weights are defined as

$$\tilde{w}_i := \hat{p}'(\theta_i | x_o), \quad w_i := \frac{\tilde{w}_i}{\sum_{j=1}^N \tilde{w}_j} = \frac{\hat{p}'(\theta_i | x_o)}{\sum_{j=1}^N \hat{p}'(\theta_j | x_o)}. \quad (3.14)$$

Histograms and moments of the distribution can be computed using weighted samples $w_i \theta_i$. The variance of the weights can be large, implying one sample takes most of the probability mass. This occurs in the same situations when rejection sampling is inefficient.

One sampling method that can be more robust to high dimensionality and approximate posteriors that make up only a small volume (compared to the prior) is *Markov chain Monte Carlo*. Rather than explaining further, we simply define the Metropolis-Hastings acceptance probability and refer the reader to Chapter 1. Given a fixed x and proposal distribution $\mathcal{T}$ with transition density $\mathcal{T}(\theta' | \theta)$ from θ to θ' , the acceptance probability $\alpha(\theta' | \theta, x)$ in a Metropolis-Hastings step is

$$\alpha(\theta' | \theta, x_o) := \min \left(1, \frac{\hat{p}'(\theta' | x_o)}{\hat{p}'(\theta | x_o)} \frac{\mathcal{T}(\theta | \theta')}{\mathcal{T}(\theta' | \theta)} \right) = \min \left(1, \frac{\hat{p}'(\theta' | x_o) / \cancel{Z_\psi(x_o)}}{\hat{p}'(\theta | x_o) / \cancel{Z_\psi(x_o)}} \frac{\mathcal{T}(\theta | \theta')}{\mathcal{T}(\theta' | \theta)} \right) \quad (3.15)$$

This acceptance probability does not depend on $Z_\psi(x_o)$ and can therefore draw samples from the approximate posterior even when $\hat{p}'(\theta | x_o)$ is not normalised. Designing an efficient transition kernel and identifying convergence is an open area of research.

3.5.4 Estimating multiple quantities at once

Finally, some recent methods go beyond estimating the posterior, the likelihood, or the likelihood-ratio and estimate several of these quantities, typically with separate models [635, 1360, 1774]. Recently, the Simformer [636] has been proposed to estimate the full joint distribution $p(\theta, x)$ as well as all of its conditionals and marginals in a single model, based on transformers and a diffusion model. Thereby, the Simformer improves the flexibility of SBI and it combines desirable features of NPE (e.g., to directly draw samples from the posterior) and NLE (e.g., to emulate the simulator).

3.6 Current developments

Neural Posterior Estimation (NPE), as well as alternative methods such as Neural Likelihood Estimation (NLE) or Neural Ratio Estimation (NRE), have been demonstrated as powerful tools for SBI. Over the past years, researchers have extended these methods in many ways, and have developed new SBI methods that go beyond estimating the posterior or likelihood(-ratio). We outline recent trends in using neural networks for SBI.

3.6.1 Better generative models

Normalizing flows have been a popular choice for NPE for several years, likely due to their robustness in training, their ability to quickly (and exactly) evaluate the log-probability of samples, and their mass-covering property. However, the fact that the transformation T trained by normalizing flows has to be invertible can constrain the flexibility of the normalizing flow. This may reduce the accuracy of posterior estimates of normalizing flows, especially for high-dimensional and structured parameter spaces (e.g., if the parameters are a time series or an image). To overcome this, recent methods have explored other conditional generative models for NPE. In recent years, diffusion models have been particularly popular due to their ability to estimate high-dimensional parameter spaces, and to maintain the ability to evaluate the log-probability via the probability flow ODE (albeit at significantly higher computational cost) [589, 636, 1505]. Many other generative models have been explored for NPE, ranging from flow-matching [1757] to consistency models [1471], generative adversarial networks [1375], energy-based models [634], and tabular foundation models [1677]. We expect that any major advances in (conditional) generative modelling will directly carry over to improvements in NPE.

3.6.2 Improving the flexibility of simulation-based inference

A core limitation of NPE with normalizing flows is that it requires the inference task to be known upfront. If, for example, the prior changes, when the simulator is modified, or when the data representation changes, NPE has to be re-run in order to perform inference (including running simulations and training the neural network).

To overcome this, many recent methods have improved the flexibility of NPE. First, in order to perform inference given any number of i.i.d. trials, permutation-invariant embedding networks can be used, as described above [1359]. Alternatively, diffusion models enable NPE to estimate the posterior distribution given any number of i.i.d. datapoints, even when the neural network was trained on pairs of parameters and a single simulation result [589, 637, 1053]. Second, in order to enable more flexible specification of prior or simulator, it has been proposed to amortize over hyperparameters of prior or simulator [489, 1209, 1558]. In that case, the inference network is conditioned not only on data, but also on additional values describing the prior or simulator. At inference time, one can then perform inference for any prior and for any simulator. Third, in some cases and only for some observations, some parameters may be known and should not be inferred, but instead should be kept fixed [399, 636]. In other cases, data may not always be complete, and parts of an observation might be missing. In order to flexibly perform amortized inference in these cases, several methods enable the estimation of any parameter conditional of NPE and systematically deal with missing data [399, 636, 1430, 1675].

3.6.3 Improving robustness to model misspecification

A fundamental challenge for SBI emerges when the simulator is *misspecified*, i.e., when it does not match reality. In the most extreme case, the observation x_o cannot be generated by the simulator (i.e., the combination of prior and simulator) with any set of parameters and with any level of simulator noise.

This can have a drastic impact on SBI: While one might expect that, for misspecified observations, the approximate posterior will have mass in parameter regions that generate predictives that are close to the observation, or that the approximate posterior should be very uncertain, this is not the case. Indeed, neural network-based methods for SBI have been shown to react erratically to the misspecified observations and can produce non-sensible posteriors [230].

Several methods have been proposed to improve the robustness of SBI in these scenarios, including adding noise to the simulation outputs [1720], learning statistics of the observations [791] such that they are not misspecified, training on unlabelled [1187] and labelled [1731] observations when available, or changing the target of inference [573].

3.6.4 Conservative and calibrated posterior approximations

Approximate SBI posteriors can be overconfident¹, particularly when simulations are limited, motivating methods that promote conservative or calibrated uncertainty estimates [742]. Balanced neural ratio estimation modifies the NRE training objective to favour conservative approximations, while preserving its Bayes-optimal solution [404], by effectively regularising the estimate to be closer to the prior. This balancing principle has also been extended to NPE and contrastive NRE [405]. Alternatively, differentiable relaxations of coverage error can be included directly in the training objectives of amortized SBI methods, encouraging calibrated posteriors with moderate

¹This occurs when the approximate posterior is narrower than the ground truth.

additional computational cost [502]. However, expected coverage is a necessary rather than sufficient diagnostic: an approximation can exhibit nominal coverage while still differing substantially from the true posterior. Ratio coverage plots address some of these blind spots by comparing the approximate and true posteriors through an estimated density ratio [1055].

3.7 Discussion

Many processes in the natural sciences and in engineering are best described by complex, stochastic, and potentially non-differentiable simulators. For many such models, the likelihood can become excessively expensive or even impossible to evaluate. In order to perform inference with traditional Bayesian inference methods such as MCMC, scientists and engineers had to modify their model such that the likelihood can be efficiently evaluated, but which comes at the cost of reducing the fidelity of the simulator.

Simulation-based inference (SBI) makes Bayesian inference accessible to the widest class of models: It can be applied to any black-box simulator for which one can run forward simulations. In recent years, neural networks have largely improved the accuracy and applicability of SBI. These methods generate a database of parameters and corresponding simulation outputs, and then train neural networks to learn the statistical relationship between these quantities. After training, the neural network can be evaluated at any observation and enables us to infer the posterior distribution without further simulations or retraining.

SBI with neural networks has already been applied to perform inference in a wide range of disciplines in science and engineering. We expect that the continued progress in improving the accuracy, flexibility, and robustness of these approaches will have a profound impact on many fields, and we hope that it will enable new scientific discoveries.

Chapter 4

Sequential Monte Carlo samplers

This chapter develops sequential Monte Carlo (SMC) samplers as a population-based approach to inference in Bayesian deep learning. Building on the sequential Monte Carlo methods of Chapter 1, we focus on the two extensions that make them practical for neural network posteriors: multilevel SMC and scalable parallel SMC.

4.1 Motivation

Chapter 1 introduced SMC as a particle filter [446, 447]: a latent state evolving in time, tracked by a population of particles over a sequence of growing spaces. This chapter applies the same population-based approximation to a static target—a Bayesian posterior on a fixed parameter space. Without any dynamics to generate them, the intermediate distributions are now constructed by design, and the resulting methods are known as SMC samplers [402].

It is useful to place the neighbouring methods relative to this sampler. Tempering is the mechanism inside the sampler that builds the intermediate distributions, annealing the likelihood from the prior to the posterior [599, 612, 1228] (data-tempering [296] is a common alternative). Annealed importance sampling (AIS) is the reduced case in which resampling is omitted [1228], so it is best viewed as a simplified SMC sampler rather than a distinct method. The chapter’s two main contributions are developments of the SMC sampler, along two different axes: multilevel SMC (MLSMC) improves statistical efficiency by combining a hierarchy of coarse-to-fine approximations in a telescoping estimator [142, 143], while parallel SMC improves scalability by running many samplers concurrently and weighting them by their normalizing-constant estimates, avoiding the heavy communication of island particle methods [1038, 1674, 1745].

These methods matter for Bayesian deep learning because neural network posteriors are high-dimensional, multi-modal with many symmetric modes, and costly to evaluate—the regime where importance sampling degenerates and single-chain MCMC mixes slowly. An SMC sampler anneals an entire population towards the posterior, provides an unbiased estimate of the model evidence, and parallelises naturally. The remainder of the chapter develops the two extensions that make this practical at scale: MLSMC with trace-class neural network priors [252, 1488], and scalable parallel SMC [1038].

4.2 Introduction

Monte Carlo based Bayesian algorithms of the type described in Chapter 1 typically require generating many samples from a target distribution which at best can be evaluated up to a normalizing constant. This can be expensive and cumbersome in high-dimensions, and so methods to improve complexity are paramount. As discussed in Chapter 1, the best computational complexity achievable by a Monte Carlo algorithm in general is $\text{cost} \propto 1/\text{MSE}$, i.e. achieving a mean square error (MSE) of $\mathcal{O}(\varepsilon^2)$ for $\varepsilon > 0$ will asymptotically require $\mathcal{O}(\varepsilon^{-2})$ samples. This is hence referred to as the *canonical rate*, as it holds for any intractable integral for which we can generate i.i.d. unbiased samples of the integrand.

The sequential Monte Carlo (SMC) sampler [297, 355, 402] was developed at the turn of the millennium [139, 296, 629, 837, 1228], and is well suited to this high-dimensional regime. Forcing the (successive) importance distributions to be close provides a *generic mechanism* to overcome the “curse-of-dimensionality”, i.e. complexity *constant* scaling like e^P for dimension P [7, 261]. Under suitable high-dimensional stability assumptions, SMC samplers can avoid the exponential degradation of direct importance sampling and may achieve polynomial cost

scaling in dimension [141]. The SMC sampler also handles bad initialization or multi-modality on par with other population methods [997], and facilitates adaptive tuning of the Markov kernel [215], with comparable efficiency to MCMC approaches like the No-U-Turn Sampler (NUTS) [762]. These benefits are not shared by the particle filter, which does not utilize MCMC methodology. It also delivers an *unbiased* estimator of the normalizing constant, or model evidence, which can be useful in practice [297].

The simplest method for improving complexity is parallelization. Standard i.i.d. Monte Carlo simulation is “embarrassingly parallel”, as the samples can all be simulated *independently* on different machines. If the cost of each simulation dominates the $\mathcal{O}(\varepsilon^{-2})$ communication and summation operations, which should both have extremely small scalar constants which are often invisible to the algorithm ($< 10^{-9}$ smaller than likelihood computations), then the constant and hence final time complexity can be reduced significantly in practice, even down to effectively $\mathcal{O}(1)$ when the merge is invisible. Monte Carlo methods for Bayesian simulation do not typically deliver i.i.d. samples, and so they are not a priori parallel. SMC samplers admit a certain degree of parallelism in their raw form, and we will show that they can also be provably “embarrassingly” parallelized beyond this [1038].

If the underlying problem we are simulating from additionally requires a discrete approximation of a function then the complexity compounds. For example, suppose that achieving a bias² of $\mathcal{O}(\varepsilon^2)$ demands a cost of $\mathcal{O}(\varepsilon^{-\xi})$. Then the MSE complexity compounds as $\mathcal{O}(\varepsilon^{-2-\xi})$. To overcome this issue, one remedy to reduce the computational cost is through a modified Monte Carlo estimator which is known as multilevel Monte Carlo (MLMC) [627, 628, 730]. MLMC introduces a hierarchy of intermediate convergent discretization levels and constructs an estimator as a telescopic sum of a coarse and cheap estimator with many samples and successive increment estimators with increasing cost and decreasing sample size, ultimately improving the complexity to $\mathcal{O}(\varepsilon^{-(2\wedge\xi)})$ instead of the product form, which can translate to huge scaling gains for small target MSE.

MLMC methodology in the Bayesian context was first developed for Markov chain Monte Carlo (MCMC) [434, 757] and sequential Monte Carlo (SMC) samplers [142]. We consider a multilevel sequential Monte Carlo (MLSMC) sampler based on the application of a function-space version of the latter algorithm [143] to a strongly convergent class of neural network models known as *trace-class neural networks* (TNN) [252, 1488]. These are Bayesian feedforward neural networks (BNN) which impose a *width decay* on the weights instead of a standard L^2 width decay prior. This simple and natural idea eliminates exchangeability and non-identifiability of the weights and delivers strong convergence with a tunable smoothness parameter. The limiting Gaussian measures are defined in function spaces with similar properties to Sobolev spaces [181]. The resulting method was originally developed in Chada et al. [252], where the authors were able to demonstrate the canonical complexity is achievable. We will use function evaluations as the fundamental unit of computational cost, which is also a convenient proxy for wall-clock time which is agnostic to particulars of the implementation or hardware.

The organization of this chapter is as follows. First, we introduce the SMC sampler in Section 4.3. In Section 4.4 we present the MLSMC sampler method for TNN. In Section 4.5 we present the parallel SMC method.

4.3 SMC sampler

Suppose that we aim to simulate from a target distribution

$$\pi(\theta) := p(\theta \mid \mathcal{D}) \propto p(Y \mid X, \theta) p(\theta). \quad (4.1)$$

One option is to use an MCMC kernel $\mathcal{M}$ such that $p\mathcal{M} = p$ to simulate $\theta^{(k)} \sim \mathcal{M}(\theta^{(k-1)}, \cdot)$, but this method produces correlated local samples and is prone to getting stuck in a given basin of attraction. If that basin of attraction is in the set of indistinguishable basins that we want then it may not particularly matter in practice. Gradient-based MCMC methods such as HMC can reduce random-walk behaviour, but they can still mix poorly across isolated posterior modes or permutation-symmetric basins.

Population methods such as importance sampling (IS) are able to mitigate this issue, but they suffer from their own difficulties. In particular, in its raw form, IS suffers from a curse of dimensionality in both parameter and data [261], and as such it is not suitable for high dimensional problems and big data. The SMC sampler bypasses this problem by

- (i) interpolating between an easy-to-sample-from initial distribution p_0 and the target distribution of interest p with several intermediate distributions p_t such that $p_T = p$, and
- (ii) interleaving sequential importance sampling (SIS) with resampling and mutation by a sequence of appropriate MCMC kernels $\mathcal{M}_t$ such that $p_t\mathcal{M}_t = p_t$.

In the Bayesian context, typically the prior p is a suitable initial distribution. The sequence of intermediate targets can be built to gradually introduce the data, for example either from growing subsets of the data [296] or

Algorithm 10: SMC sampler (c = communication here)

Inputs : number of particles M , sequence of distributions π_t , MCMC kernels $\mathcal{M}_t$
Outputs: particle system $\{\theta_t^i\}_{i=1}^M$ and normalizing constant estimate Z_T^M
initialize $\theta_0^i \sim \pi_0$ for $i = 1, \dots, M$; set $Z_0^M \leftarrow 1$
for $t = 1$ **to** T (*in serial*) **do**
 // (Optional) adapt annealing parameter
 select λ_t such that $\text{ESS} = \alpha M$
 // (c) Update estimate of normalizing constant
 $Z_t^M \leftarrow Z_{t-1}^M \cdot \frac{1}{M} \sum_{k=1}^M p(Y \mid \theta_{t-1}^k, X)^{\lambda_t - \lambda_{t-1}}$
 for $i = 1$ **to** M (*in parallel*) **do**
 define $w_t^i \propto p(Y \mid \theta_{t-1}^i, X)^{\lambda_t - \lambda_{t-1}}$
 // (c) Selection step
 sample ancestor index $I_t^i \sim \{w_t^1, \dots, w_t^M\}$
 // Mutation step
 sample $\theta_t^i \sim \mathcal{M}_t(\theta_{t-1}^{I_t^i}, \cdot)$

with a tempering schedule $0 = \lambda_0 < \lambda_1 < \dots < \lambda_T = 1$ [1228]:

$$p_t(\theta) \propto p(Y \mid X, \theta)^{\lambda_t} p(\theta).$$

Three quantities are tracked along this path. The normalizing constant at temperature λ_t is

$$Z_t = \int p(Y \mid X, \theta)^{\lambda_t} p(\theta) d\theta,$$

which at $\lambda_T = 1$ equals the model evidence $p(Y \mid X)$. Algorithm 10 maintains an unbiased estimate Z_t^M at each step, with Z_T^M returned as the estimate of the evidence. Successive targets are linked by the incremental importance weight that reweights particle i from π_{t-1} towards π_t ,

$$w_t^i \propto p(Y \mid X, \theta_{t-1}^i)^{\lambda_t - \lambda_{t-1}}, \quad \tilde{w}_t^i = \frac{w_t^i}{\sum_{k=1}^M w_t^k}.$$

How evenly these weights are spread is measured by the effective sample size (ESS),

$$\text{ESS} = \frac{1}{\sum_{i=1}^M (\tilde{w}_t^i)^2} \in [1, M].$$

Values near M indicate balanced weights, whereas small values mean a few particles dominate. The ESS drives the algorithm in two ways: a low value motivates resampling, and in the adaptive variant the next temperature is chosen to preserve a target $\text{ESS} = \alpha M$. Data-tempering and likelihood-tempering construction schemes for the intermediate targets can also be mixed and matched. We will consider another option in Section 4.4. See Algorithm 10 for the version with tempering for Bayesian inference. The steps which require communication during execution are denoted with (c). It is noteworthy that if we skip the resampling step then the algorithm is called annealed importance sampling (AIS) and was introduced in Neal [1228]. The normalizing constant then only needs to be computed once at the end. This version of the algorithm is convenient because it is embarrassingly parallel – communication is only required at resampling times. However, the purpose of resampling is to prevent degeneracy and improve stability, and this benefit often outweighs the added variance [296, 297].

4.4 MLSMC for TNN

In this section we will build up the MLSMC for TNN method. First we introduce the MLMC method, followed by the TNN. Then we present the MLSMC sampler and provide theoretical justification, and numerical results validating and extending the theory.

4.4.1 MLMC method

We shall begin with a short review of MLMC. MLMC is useful when accurate simulations are expensive but coarse approximations are cheap. Rather than estimating the finest-level expectation directly, one estimates a coarse expectation plus a sequence of corrections. If consecutive levels are strongly coupled, the correction variances decay rapidly, so fewer samples are needed at expensive fine levels. Let us assume that we are given a probability density p , on a state-space $\mathbf{U}$. It is of interest to compute expectations of p -integrable functions, $\varphi : \mathbf{U} \rightarrow \mathbb{R}$; $p(\varphi) := \int_{\mathbf{U}} \varphi(u)p(u)du$. Now, we assume that we need to approximate p by a density p_l on a state-space $\mathbf{U}_l \subseteq \mathbf{U}$ such that:

1. $\lim_{l \rightarrow \infty} p_l(\varphi) = p(\varphi)$, for any integrable $\varphi : \mathbf{U} \rightarrow \mathbb{R}$.
2. The cost of computing with p_l is increasing in l .

Let $L \in \{1, 2, \dots\}$ be given, we have the telescoping sum identity

$$p_L(\varphi) = p_0(\varphi) + \sum_{l=1}^L [p_l - p_{l-1}](\varphi), \quad (4.2)$$

with $[p_l - p_{l-1}](\varphi) = p_l(\varphi) - p_{l-1}(\varphi)$. The idea behind MLMC is to leverage this identity to reduce the complexity of the approximation. This is achieved by constructing *couplings* of (p_l, p_{l-1}) , $l \in \{1, \dots, L\}$: joint densities $\check{p}_l$, on $\mathbf{U}_l \times \mathbf{U}_{l-1}$ such that $\int_{\mathbf{U}_l} \check{p}_l(u_l, u_{l-1}) du_l = p_{l-1}(u_{l-1})$ and $\int_{\mathbf{U}_{l-1}} \check{p}_l(u_l, u_{l-1}) du_{l-1} = p_l(u_l)$.

The MLMC estimator can be constructed in the following way.

1. Sample $U_0^1, \dots, U_0^{M_0}$ i.i.d. from p_0 , with $M_0 \in \mathbb{N}$ given.
2. For $l \in \{1, \dots, L\}$ independently of all other random variables, sample $(U_l^1, \tilde{U}_{l-1}^1), \dots, (U_l^{M_l}, \tilde{U}_{l-1}^{M_l})$ i.i.d. from $\check{p}_l$, with $M_l \in \mathbb{N}$ given.

Then, we have the approximation

$$p_L^{ML}(\varphi) := \frac{1}{M_0} \sum_{i=1}^{M_0} \varphi(U_0^i) + \sum_{l=1}^L \frac{1}{M_l} \sum_{i=1}^{M_l} \{\varphi(U_l^i) - \varphi(\tilde{U}_{l-1}^i)\}. \quad (4.3)$$

Using i.i.d. samples $U_L^1, \dots, U_L^M$ from p_L to approximate $p_L(\varphi)$, we have

$$p_L^{IID}(\varphi) := \frac{1}{M} \sum_{i=1}^M \varphi(U_L^i). \quad (4.4)$$

Both estimators are unbiased, so we have the general decomposition of an unbiased estimator $\hat{p}$

$$\mathbb{E}[(p_L^{ML}(\varphi) - p(\varphi))^2] = \underbrace{\mathbb{V}\text{ar}[p_L^{ML}(\varphi)]}_{\text{variance}} + \underbrace{[p_L - p](\varphi)^2}_{\text{bias}^2}, \quad (4.5)$$

with $\mathbb{V}\text{ar}$ representing the variance operator. The bias is equal for both, and to optimize the complexity this should match the variance. The variance of (4.3) is

$$\mathbb{V}\text{ar}[p_L^{ML}(\varphi)] = \frac{\mathbb{V}\text{ar}[\varphi(U_0^1)]}{M_0} + \sum_{l=1}^L \frac{\mathbb{V}\text{ar}[\varphi(U_l^1) - \varphi(\tilde{U}_{l-1}^1)]}{M_l}, \quad (4.6)$$

whereas for (4.4)

$$\mathbb{V}\text{ar}[p_L^{IID}(\varphi)] = \frac{\mathbb{V}\text{ar}[\varphi(U_L^1)]}{M}.$$

Now, if the couplings $\check{p}_l$ are constructed so that $\mathbb{V}\text{ar}[\varphi(U_l^1) - \varphi(\tilde{U}_{l-1}^1)]$ falls sufficiently quickly with l then it can be possible to achieve an MSE (as in (4.5)) for the estimator (4.3) which is of the same order as (4.4) except for a cost that is lower. This is characterized in the following result.

Theorem 4.1 (Giles [627]). *Suppose that there exists constants $(\alpha, \beta, \gamma) \in \mathbb{R}_+^3$ with $\alpha \geq \frac{\min(\beta, \gamma)}{2}$ such that*

$$(i) \quad |p_l(\varphi) - p(\varphi)| = \mathcal{O}(2^{-\alpha l}).$$

$$(ii) \text{ Var}[\varphi(U_l^1) - \varphi(U_{l-1}^1)] = \mathcal{O}(2^{-\beta l}).$$

$$(iii) C_l = \mathcal{O}(2^{\gamma l}).$$

Then for any $\varepsilon < 1$ and $L := \lceil \log(1/\varepsilon) \rceil$, there exists $(M_1, \dots, M_L) \in \mathbb{N}^L$ such that

$$\text{MSE} = \mathbb{E}[(p_L^{ML}(\varphi) - p(\varphi))^2] = \mathcal{O}(\varepsilon^2), \quad (4.7)$$

and

$$\text{Cost(MLMC)} := \sum_{l=0}^L M_l C_l = \begin{cases} \mathcal{O}(\varepsilon^{-2}), & \text{if } \beta > \gamma, \\ \mathcal{O}(\varepsilon^{-2}(\log \varepsilon)^2), & \text{if } \beta = \gamma, \\ \mathcal{O}(\varepsilon^{-2 - \frac{(\gamma - \beta)}{\alpha}}), & \text{if } \beta < \gamma, \end{cases} \quad (4.8)$$

where C_l is the cost for one sample of $\varphi(U_l^i) - \varphi(U_{l-1}^i)$.

Asymptotically the cost in (4.8) is smaller than that of using the estimator (4.4). It is simple to see that the corresponding optimal complexity of the single level estimator (4.4) is given by the product of $\mathcal{O}(\varepsilon^{-2})$ samples at a per sample complexity of $\mathcal{O}(\varepsilon^{-\gamma/\alpha})$, i.e. $\mathcal{O}(\varepsilon^{-2-\gamma/\alpha})$, which is larger than the MLMC complexity for $\beta > 0$. The case of $\beta > \gamma$ is referred to as the canonical rate of convergence, i.e. $\mathcal{O}(\varepsilon^{-2})$, in other words this is the best rate that one can obtain.

4.4.2 Trace class neural networks

First we set up the problem. Suppose we have data $\mathcal{D} = ((x_1, y_1), \dots, (x_N, y_N))$, $N \in \mathbb{N}$, where, for $i \in \{1, \dots, N\}$, $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$. The objective is to infer a predictive model $f : \mathbf{X} \rightarrow \mathbf{Y}$ based on the data. One way to do this is with a parametric model of the form $f : \mathbf{X} \times \Theta \rightarrow \mathbf{Y}$, with $\Theta \subseteq \mathbb{R}^P$. It is assumed that $x_{1:N}$ are deterministic, i.e. the discriminative vs. generative view of supervised learning.

Regression. If $\mathbf{Y} = \mathbb{R}^C$, then we will assume that for $i \in \{1, \dots, N\}$

$$y_i = f(x_i, \theta) + \epsilon_i, \quad \epsilon_i \stackrel{\text{ind}}{\sim} \mathcal{N}_C(0, \Sigma_i), \quad (4.9)$$

where ind denotes independence across the indices $i \in \{1, \dots, N\}$ and $\mathcal{N}_C(\mu, \Sigma)$ denotes the C -dimensional Gaussian distribution with mean μ and covariance matrix Σ . The likelihood function in this case is

$$p(Y|\theta, X) = \prod_{i=1}^N \phi_C(y_i; f(x_i, \theta), \Sigma_i), \quad (4.10)$$

where $\phi_C(y; \mu, \Sigma)$ is the C -dimensional Gaussian density of mean μ , covariance Σ evaluated at y given.

Classification. If $\mathbf{Y} = \{1, \dots, C\}$ for some $C \in \mathbb{N}$, for convenience of the present exposition we will define $f : \mathbf{X} \times \Theta \rightarrow \mathbb{R}^C$, with $f(x, \theta) = (f_1(x, \theta), \dots, f_C(x, \theta))$, and then separately the *softmax* function as

$$S_k(x_i, \theta) := \frac{\exp\{f_k(x_i, \theta)\}}{\sum_{j=1}^C \exp\{f_j(x_i, \theta)\}}, \quad k \in \mathbf{Y}. \quad (4.11)$$

We will assume that $y_i \sim S(x_i, \theta)$, independently for $i \in \{1, \dots, N\}$, where $S(x, \theta) = (S_1(x, \theta), \dots, S_C(x, \theta))$ denotes a categorical distribution on C outcomes, given input x . In this case, the likelihood is

$$p(Y|\theta, X) = \prod_{i=1}^N \prod_{k=1}^C S_k(x_i, \theta)^{\mathbb{I}_{[y_i=k]}}. \quad (4.12)$$

The predictive S is Lipschitz, so conclusions about f translate immediately. We conclude this part by stating the key advantages of TNNs: one is able to have a stable infinite limit related to the width, as well as high-dimensional scalability, due to the fact that one does not need to define the covariance operator structure.

Bayesian neural networks

Define the element-wise activation function as σ . Let $\mathbf{X} = \mathbb{R}^D$ and $\mathbf{Y} = \mathbb{R}^C$. The DNN can be defined in the following way. Let $D \in \mathbb{N}$, $(D_0, \dots, D_L) \in \mathbb{N}^{L+1}$ be given, with the constraint that $D_0 = D$ for the input layer and $D_L = C$ for the output layer. Now for weights $A_d \in \mathbb{R}^{D_{L'} \times D_{L'-1}}$ and biases $b_d \in \mathbb{R}^{D_d}$, $d \in \{1, \dots, L\}$ also given, we use the notation $\theta := ((A_1, b_1), \dots, (A_L, b_L))$ and so $\theta \in \Theta = \bigotimes_{L'=1}^L \{\mathbb{R}^{D_d \times D_{d-1}} \times \mathbb{R}^{D_d}\}$. Now set

$$\begin{aligned} g_0(x, \theta) &:= A_1 x + b_1, \\ g_{L'}(x, \theta) &:= A_{L'} \sigma(g_{L'-1}(x)) + b_{L'}, \quad L' \in \{1, \dots, L-1\}, \\ f(x, \theta) &:= A_L \sigma(g_{L-1}(x)) + b_L, \end{aligned} \quad (4.13)$$

where $f(x, \theta)$ is the output of the final layer of the DNN. The BNN is given by placing a prior p on Θ .

TNN

We now introduce the trace class neural network (TNN) priors, which were first proposed in Sell and Singh [1488], and differ from standard BNN priors in two fundamental ways. The first is that the prior on the weights and biases depends on the rows and columns, as opposed to standard isotropic weight decay. This is referred to as *width-decay* and it eliminates exchangeability and the associated non-identifiability. The second thing is that nested approximations of the well-defined TNN function-space converge strongly to a non-Gaussian process in the limit of infinite width. This stands in contrast to standard isotropic priors, under which the appropriately rescaled network converges weakly to a Gaussian process [386, 1226], so that the infinite-width limit is a kernel method and the compositional structure of the prior is lost. This will be made precise below.

These priors were introduced to mimic Gaussian measure priors $p \sim \mathcal{N}(0, \mathcal{C})$ for inverse problems over function-space, where a common way to simulate Gaussian random fields is to use the Karhunen-Loève expansion

$$f_{\text{KL}} = \sum_{j \in \mathbb{Z}^+} \sqrt{\lambda_j} \iota_j \Phi_j, \quad \iota_j \sim \mathcal{N}(0, 1), \quad (4.14)$$

where $(\lambda_j, \Phi_j)_{j \in \mathbb{Z}^+}$ are the associated eigenbasis of the covariance operator $\mathcal{C}$, and $\{\iota_j\}_{j \in \mathbb{Z}^+}$ is Gaussian white noise. For a detailed description of the derivation of (4.14) and its application to stochastic numerical problems, we refer the reader to Lord et al. [1078]. An issue that can arise with using (4.14) is that these priors do not scale well with input dimension, which is the initial motivation for TNN priors, where the variances of the weights and biases are summarized in a trace-class diagonal covariance operator $\mathcal{C}$. See [1488, Theorem 1]. More precisely, the TNN prior is given by

$$A_{i,j,d} \sim \mathcal{N}(0, (ij)^{-s}), \quad b_{i,d} \sim \mathcal{N}(0, i^{-s}). \quad (4.15)$$

The tuning parameter s controls how much information one believes concentrates on the first nodes, and as such also controls the smoothness, expressiveness, and flexibility of the prior in terms of the functions it can represent. In the case of $s > 1$, we refer to the prior as trace-class, which is the motivation for the name trace-class neural network prior.

A convenient property of these priors is strong convergence, which is made precise with the following proposition. The proof is given in Chada et al. [252]. This is the key result that enables application of the MLSMC algorithm to TNN. The advantage of TNN is that it allows one not to prespecify the covariance operator structure in the prior, and it is also capable of having an infinite-width limit.

Proposition 4.1. *Assume that for all $z \in \mathbb{R}$, $|\sigma(z)| \leq |z|$. Then for $s > 1/2$, the prior predictive output is square summable for the prior defined as in (4.15). Furthermore, let $x \in \mathbf{X}$, and consider the TNN $f_l(x, \theta_l)$ truncated at $D_l = 2^l$ width, for $l \in \mathbb{N}$, with the limit denoted $f(x, \theta) = \lim_{l \rightarrow \infty} f_l(x, \theta_l)$, as described above. Then there is a $K(x) > 0$ such that*

$$\mathbb{E} [|f_l(x, \theta_l) - f(x, \theta)|^2] \leq K 2^{-(2s-1)l}. \quad (4.16)$$

Multilevel TNN

We now consider the question of choosing the dimension of θ , in particular $D_{L'}$ for $L' \in \{1, \dots, L-1\}$ (since D_0 and D_L are fixed to the input and output dimensions). For simplicity, here we assume L is fixed and the NN width is the same for all layers other than the input or the output, i.e. $D_{L'} = D_{L''}$ for $L', L'' \in \{1, \dots, L-1\}$, but dependent upon some resolution parameter $l \in \mathbb{N}$. We can now re-define $D_l = 2^l$, noting that $D_0 = D$ and $D_L = C$, so those variables are no longer needed. Denote the corresponding vector of parameters by $\theta_l :=$

$((A_1^l, b_1^l), \dots, (A_D^l, b_D^l)) \in \Theta_l \subset \Theta$. The limiting infinite width NN output function $f(x, \theta)$ (4.13) approximated at finite resolution l is denoted by $f_l(x, \theta_l)$, the likelihood by $p_l(Y|\theta_l, X)$, and the posterior distribution by

$$\pi_l(\theta_l) := p_l(\theta_l|\mathcal{D}) \propto p_l(Y|\theta_l, X)p_l(\theta_l) =: \kappa_l(\theta_l). \quad (4.17)$$

We view this posterior (and the corresponding $f_l(x, \theta_l)$) as a finite approximation of the posterior associated to the non-parametric limiting DNN as $l \rightarrow \infty$, assuming it exists.

4.4.3 MLSMC TNN method

We begin this section by firstly presenting our algorithm for approximating functional expectations, with respect to the posterior π , such as the posterior predictive $\mathbb{E}_\pi[f(x, \theta)]$, as well as some mathematical results which justify their implementation. The form of our results are such that they extend trivially to objective functions $\varphi \circ f$, for Lipschitz φ , which includes most conventional performance metrics, such as negative log likelihood and accuracy. The results will be presented first for the posterior predictive, and then extended as a corollary. Noting that $\mathbb{E}_\pi[f(x, \theta)] = \mathbb{E}_{\pi_L}[f_L(x, \theta_L)] + (\mathbb{E}_\pi[f(x, \theta)] - \mathbb{E}_{\pi_L}[f_L(x, \theta_L)])$, our main objective is to leverage the MLMC machinery for the following telescopic identity:

$$\mathbb{E}_{\pi_L}[f_L(x, \theta_L)] = \sum_{l=1}^L \{ \mathbb{E}_{\pi_l}[f_l(x, \theta_l)] - \mathbb{E}_{\pi_{l-1}}[f_{l-1}(x, \theta_{l-1})] \} + \mathbb{E}_{\pi_0}[f_0(x, \theta_0)]. \quad (4.18)$$

We will show that the optimal computational cost for achieving a MSE equal to $(\mathbb{E}_\pi[f(x, \theta)] - \mathbb{E}_{\pi_L}[f_L(x, \theta_L)])^2$ by approximating the individual summands on the R.H.S. independently is lower than directly approximating $\mathbb{E}_{\pi_L}[f_L(x, \theta_L)]$.

Algorithm

The approach we construct follows that in Beskos et al. [142, 143] and in order to directly apply some of its results, we introduce additional notation. Throughout this exposition $(x, L) \in \mathbf{X} \times \mathbb{N}$ are fixed and given. Define the spaces constructively as $\Theta_l = \Theta_{l-1} \times \Delta_l$, for $l \in \{1, \dots, L\}$, and $\Delta_0 = \Theta_0$; the significance of these spaces will be made clear below. Set $\theta_0 = \delta_0 \in \Theta_0$ and for $l \in \{1, \dots, L\}$

$$\theta_l = (\theta_{l-1}, \delta_l) = (\delta_0, \dots, \delta_l) \in \Theta_l. \quad (4.19)$$

The quantities $(\delta_1, \dots, \delta_l)$ will be used to denote the increase in dimension between θ_{l-1} and θ_l say, which is pivotal to the algorithm that we will present. Let $q_l(\cdot|\theta_{l-1})$ be a positive probability density on Δ_l and $q_0(\theta_0)$ a positive probability density on Θ_0 . Set $G_0(\theta_0) = \kappa_0(\theta_0)/q_0(\theta_0)$ and define weights as

$$G_l(\theta_l) = \frac{\kappa_l(\theta_l)}{\kappa_{l-1}(\theta_{l-1})q_l(\delta_l|\theta_{l-1})}. \quad (4.20)$$

Let $(\mathcal{K}_l)_{l \in \{1, \dots, L-1\}}$ be a sequence of $(\pi_l)_{l \in \{1, \dots, L-1\}}$ -invariant Markov kernels and define for $l \in \{1, \dots, L-1\}$

$$\mathcal{M}_l(\theta_l, d\theta'_{l+1}) = \mathcal{K}_l(\theta_l, d\theta'_l)q_{l+1}(\delta_{l+1}|\theta'_l)d\delta_{l+1}, \quad (4.21)$$

where $\theta'_{l+1} = (\theta'_l, \delta_{l+1})$ and $d\delta_l$ is the appropriate dimensional Lebesgue measure. This kernel is invoked in Step 4 of Algorithm 11.

Set $\eta_0(\theta_0) = q_0(\theta_0)$ and for $l \in \{1, \dots, L\}$

$$\eta_l(\theta_l) = \pi_{l-1}(\theta_{l-1})q_l(\delta_l|\theta_{l-1}). \quad (4.22)$$

The algorithm to be presented will provide a sample-based approximation of these probability densities and in particular, associated expectations. In the subsequent exposition, given $(\theta_l^1, \dots, \theta_l^M) \in \Theta_l^M$, the so-called M -empirical measure will be denoted η_l^M . In other words we will have access to $M \in \mathbb{N}$ samples $(\theta_l^1, \dots, \theta_l^M) \in \Theta_l^M$ so that

$$\eta_l^M(\varphi_l) := \frac{1}{M} \sum_{i=1}^M \varphi_l(\theta_l^i) \rightarrow \eta_l(\varphi_l) \quad \text{a.s.} \quad (4.23)$$

Algorithm 11: Multilevel Sequential Monte Carlo Sampler for TNN

Inputs : highest resolution $L \in \mathbb{N}$; number of samples per level $(M_0, \dots, M_L) \in \mathbb{N}^{L+1}$, with $+\infty > M_0 \geq M_1 \geq \dots \geq M_L \geq 1$

Outputs: particles $(\theta_1^1, \dots, \theta_1^{M_1}, \dots, \theta_L^1, \dots, \theta_L^{M_L})$, from which (4.27) is constructed

// Initialization

for $i = 1$ **to** M_0 **do**

 independently sample $\theta_0^i \sim q_0(\theta_0)d\theta_0$

$l \leftarrow 0$

// Iteration across levels

while $l < L$ **do**

for $i = 1$ **to** M_{l+1} **do**

 sample $\theta_{l+1}^i \mid \theta_l^1, \dots, \theta_l^{M_l}$ independently using

$$\sum_{j=1}^{M_l} \frac{G_l(\theta_l^j)}{\sum_{s=1}^{M_l} G_l(\theta_l^s)} \mathcal{M}_l(\theta_l^j, d\theta_{l+1})$$

$l \leftarrow l + 1$

Recalling (4.18), we will approximate the increments as follows

$$\frac{\eta_l^{M_l}(G_l f_l)}{\eta_l^{M_l}(G_l)} - \eta_l^{M_l}(f_{l-1}). \quad (4.24)$$

Note that $\eta_l^{M_l}(f_{l-1})$ will converge in probability (as $M_l \rightarrow \infty$) to $\pi_{l-1}(f_{l-1})$ [401]. Then $\eta_l^{M_l}(G_l)$ will converge to

$$\int_{\Theta_l} \pi_{l-1}(\theta_{l-1}) q_l(\delta_l \mid \theta_{l-1}) \frac{\kappa_l(\theta_l)}{\kappa_{l-1}(\theta_{l-1}) q_l(\delta_l \mid \theta_{l-1})} d\theta_l = \frac{1}{Z_{l-1}} \int_{\Theta_l} \kappa_l(\theta_l) d\theta_l = \frac{Z_l}{Z_{l-1}}, \quad (4.25)$$

and $\eta_l^{M_l}(G_l f_l)$ converges to

$$\int_{\Theta_l} \pi_{l-1}(\theta_{l-1}) q_l(\delta_l \mid \theta_{l-1}) \frac{\kappa_l(\theta_l)}{\kappa_{l-1}(\theta_{l-1}) q_l(\delta_l \mid \theta_{l-1})} f_l(x, \theta_l) d\theta_l = \frac{1}{Z_{l-1}} \int_{\Theta_l} \kappa_l(\theta_l) f_l(x, \theta_l) d\theta_l. \quad (4.26)$$

Consequently, one can use the following approximation of $\pi_L(f_L)$:

$$\hat{\pi}_L(f_L) = \sum_{l=1}^L \left\{ \frac{\eta_l^{M_l}(G_l f_l)}{\eta_l^{M_l}(G_l)} - \eta_l^{M_l}(f_{l-1}) \right\} + \frac{\eta_0^{M_0}(G_0 f_0)}{\eta_0^{M_0}(G_0)}. \quad (4.27)$$

Ordinarily we choose q_l so that for each $\theta_l \in \Theta_l$

$$p_l(\theta_l) = p_{l-1}(\theta_{l-1}) q_l(\delta_l \mid \theta_{l-1}). \quad (4.28)$$

This means that

$$G_l(\theta_l) = \frac{p_l(Y \mid \theta_l, X)}{p_{l-1}(Y \mid \theta_{l-1}, X)}. \quad (4.29)$$

The main result is given in Chada et al. [252].

Proposition 4.2. *Under suitable assumptions, an analog of Theorem 4.1 applies to MLSMC with rates determined by (4.16) for costs $C_l \propto 2^{2l}$, i.e. our effective $2\alpha = \beta = 2s - 1$ and $\gamma = 2$. Hence we are able to choose $L, \{M_1, \dots, M_L\}$ appropriately to achieve MSE of $\mathcal{O}(\varepsilon^2)$ for the canonical complexity $\mathcal{O}(\varepsilon^{-2})$.*

4.4.4 Numerical experiments

Regression Problem

The first numerical experiment will be based on a well-specified Bayesian regression problem, as given in (4.9), i.e. the data is generated from the model. We let $\Sigma_i^2 = 0.01^2 I$, $\sigma(z) = \tanh(z)$, $D = 10$, $D_l = 2^l$, and $N = 200$.

The inputs are simulated as $x_i \sim \mathcal{N}(2, 0.5)$. We use a high-resolution parameter $l = 9$ for the ground truth. We compare SMC and MLSMC samplers, using 100 replications to compute MSE. We report the complexity rate ξ such that $\text{cost} \propto \text{MSE}^{-\xi}$ for different values of $\beta = 2s - 1$. Recall that $C_l = \mathcal{O}(2^{2l})$ so one expects to attain the canonical rate of convergence $\xi = 1$ when $s > 1.5 \rightarrow \beta > 2$. We conduct our numerical experiment with levels $L \in \{3, 4, \dots, 7\}$ and smoothness parameters $s \in \{3, 1.1\}$, delivering canonical and sub-canonical convergence behaviour. The results are presented in Figure 4.1, including credible sets around the MSE values, given by the thin blue and red curves. Further details on implementation of the MLSMC sampler in general can be found in Beskos et al. [143].

The MLSMC sampler shows computational benefits, where for a given MSE the cost associated with attaining that order of MSE is smaller than the SMC sampler. We can see that (i) the rates are different, with MLSMC approximately achieving the canonical $1/\text{MSE}$ for $s = 3$ and (ii) the difference in cost for the lowest MSE is approximately a factor of 10 for $s = 3$, so the asymptotic benefit is achievable at a finite resolution, and (iii) that benefit is smaller in the sub-canonical case $s = 1.1$.

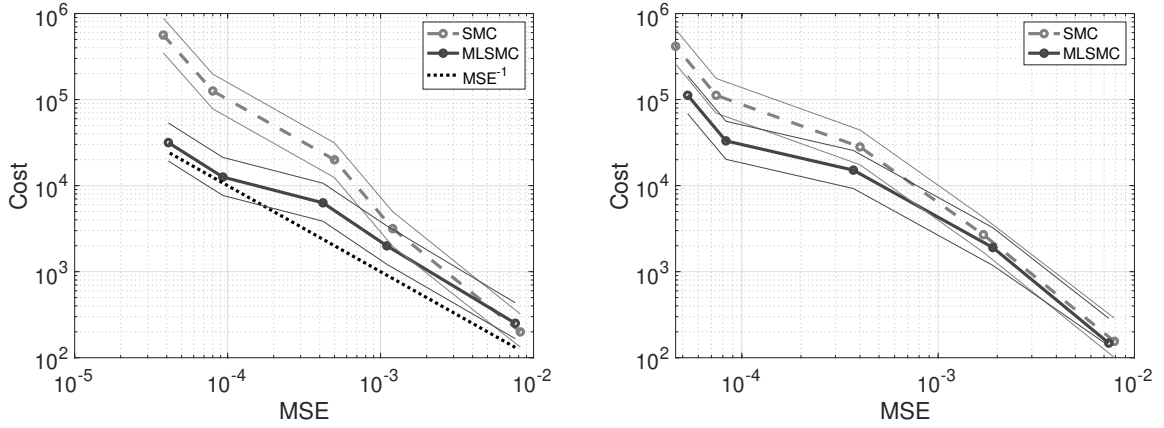


Figure 4.1: Regression problem: error vs cost plots for SMC and MLSMC using TNN priors. Left: $s = 3$. Right: $s = 1.1$. Credible intervals are provided by thin curves.

Binary MNIST Classification

As a second proof of concept, we consider a simplified binary MNIST [995] classification task using $D = 100$ principal components (PCs) of the original 28×28 image dataset, with $N = 400$. The results are similar to the regression case, and are presented in Figure 4.2.

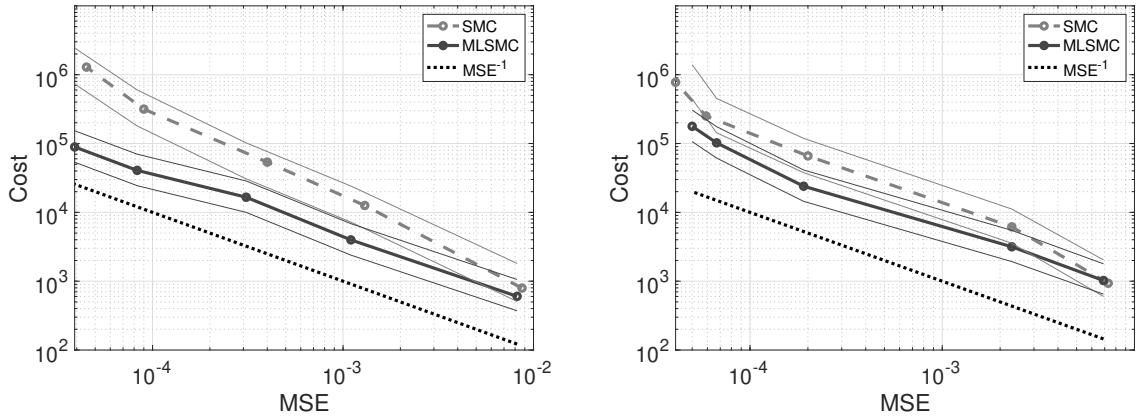


Figure 4.2: MNIST Classification problem: error vs cost plots for SMC and MLSMC, using TNN priors. Left: $s = 3$. Right: $s = 1.1$. Credible sets are provided in the thin curves.

4.5 Parallel SMC

For this section, we will assume a fixed and finite network architecture f of the standard form, as introduced in Section 4.4.2. This gives rise to a computable target distribution (up to a normalizing constant) $\pi = \kappa/Z$.

SMC is inherently parallel, as the MCMC moves for each particle can be executed concurrently [997, 1282, 1581]. This is where the likelihood computations happen, which are the most expensive and dominant contributors to the computational complexity. One must take care that sufficient memory per core is available or else this approach will hit the “memory wall” [822]. The resampling stage requires communication between all particles, which could also be a bottleneck depending on the communication bandwidth, but this may not be an issue for single-instruction-multiple-data (SIMD)-type architectures as found on a single multi-core CPU or GPU. This is the most common approach for leveraging parallelism in SMC, as it is very simple to implement and can sometimes deliver strong parallel scaling [997].

Algorithm 12: SMC_{||} sampler

Inputs : number of parallel runs R ; number of particles M

Outputs: particle systems $\{\theta^{i,r}\}_{i=1}^M$ and normalizing constants $Z^{M,r}$ for $r = 1, \dots, R$

for $r = 1$ **to** R **do**

 run Algorithm 10

 output $\{\theta^{i,r}\}_{i=1}^M$ and $Z^{M,r}$

Beyond this inter-SMC parallelism, the combination of parallel SMCs has recently been considered in Vergé et al. [1674], Whiteley et al. [1745]. The recommended approach typically involves communication between *all samples*, which improves stability but hinders scalability for large models which need to be distributed across many SIMD nodes, which may either be disconnected or have slower between-node inter-connect. The island particle model [1674] separates the total number of samples M into $R \leq R_{\max}$ SMC islands with M/R samples each. Without between-SMC interaction, naive (equal-weight) averaging results in an asymptotic bias penalty $(R/M)^2$ [346, 1674]. However, *by assigning appropriate weights to each SMC we can eliminate this penalty* [355, 1745]. The resulting scalable parallel SMC sampler is denoted by SMC_{||}.

Algorithm 12 displays the SMC_{||} method, and we define the consistent (in R , for finite M suitably large) SMC_{||} ratio estimator as

$$\hat{\varphi}_{\text{SMC}_{||}} = \sum_{r=1}^R \omega_r \pi^{M,r}(\varphi), \quad \omega_r = \frac{Z^{M,r}}{\sum_{s=1}^R Z^{M,s}}. \quad (4.30)$$

This estimator is consistent, which is not the case for a naive unweighted average. See Proposition 4.3 and the discussion above.

4.5.1 Parallel MCMC

MCMC can also exploit parallel resources: one can retain a single sample from each of M parallel chains, each run for b warm-up epochs, achieving $\text{MSE} = O(e^{-b} + 1/M)$ [1131], which is indistinguishable from the single-chain result for $b \gg \log M$, at a parallel cost equivalent to SMC when the integrated autocorrelation time matches the total number of mutation epochs. However, unlike SMC, the resulting estimator is not consistent for finite b , which can potentially spoil convergence.

4.5.2 Theoretical Result

The convergence result is made rigorous in the following proposition. Suitable assumptions and proof are provided in Liang et al. [1038].

Proposition 4.3. *For suitable φ, m, M, T , there exists a $K > 0$, which depends on φ, m, T , such that for any $R \in \mathbb{N}$,*

$$\mathbb{E}[(\hat{\varphi}_{\text{SMC}_{||}} - \pi(\varphi))^2] \leq \frac{K}{MR}. \quad (4.31)$$

4.5.3 Numerical experiments

As above, we will measure wall-clock time complexity in terms of function evaluations, as an implementation and hardware agnostic proxy. We will refer to a likelihood evaluation as an *epoch*, in analogy with stochastic

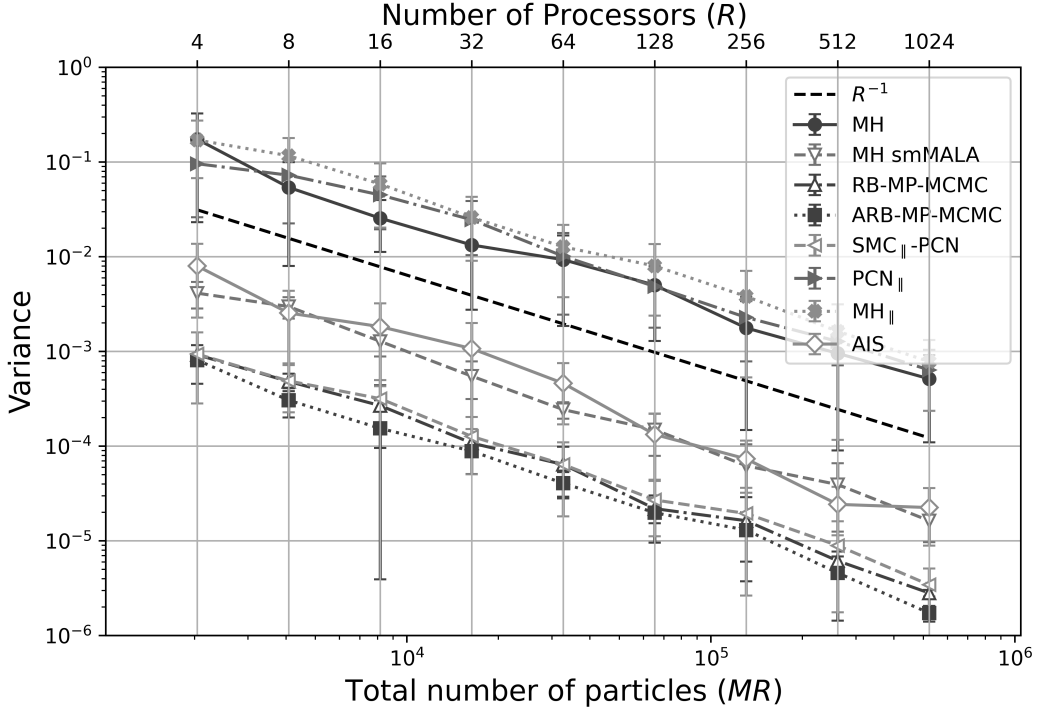


Figure 4.3: Empirical variance of posterior mean estimates for Bayesian logistic regression on Australian Credit ($m = 690$, $d = 15$), for $\text{SMC}_{\parallel}\text{-pCN}$, various $\text{MCMC}_{\parallel}$ methods, and AIS. $\text{SMC}_{\parallel}\text{-pCN}$ and the $\text{MCMC}_{\parallel}$ baselines are compared at matched total sample size MR , with $M = 511$ particles per process; AIS is equivalent to $M = 1$ particle per process. $\text{SMC}_{\parallel}\text{-pCN}$ uses pCN mutations with 30 mutation steps per tempering stage and adaptive tempering that keeps the effective sample size above $M/2$. Curves are averaged over 25 repetitions; vertical bars show ± 3 standard errors. Both axes are log-scaled.

gradient descent (SGD) [194, 1404] based optimization methods like Adam [894]. However, note that the time to compute an epoch of mini-batches depends heavily on the size of the problem and the details of the hardware and implementation. If all data fits on a single computational unit (either a node or a core), then a serial epoch of mini-batch likelihood computations will typically require much longer wall-clock time in comparison to a single likelihood evaluation. In contrast, if only a single mini-batch fits on a node, and interconnect is slow, then even data parallel may be slower than serial mini-batches. It is also worth noting that mini-batch gradients can deliver comparable gain to full gradient steps far from convergence, which can significantly accelerate progress as measured by epochs [195].

Proof of concept

First, we will consider logistic regression on the Australian Credit data [1355], illustrating comparable results to state-of-the-art (SOTA) parallel MC algorithms and the $1/R$ convergence guaranteed by Proposition 4.3. The dataset has $d = 15$ covariates and $m = 690$ data and Bayesian logistic regression is used. We compare $\text{SMC}_{\parallel}\text{-pCN}$ (preconditioned Crank–Nicolson, pCN) with annealed importance sampling [1228], various $\text{MCMC}_{\parallel}$ methods, and recent synchronous-parallel MCMC [1479]. Figure 4.3 shows the empirical variance for posterior mean estimates of the proposed methods with respect to the number of samples and number of parallel processes, consistent with the $1/R$ convergence result of Proposition 4.3. $\text{SMC}_{\parallel}\text{-pCN}$ attains the lowest empirical variance among the baselines shown in Figure 4.3 that do not require communication scaling with R . The $\text{MCMC}_{\parallel}$ data in Figure 4.3 is reproduced directly from Schwedes and Calderhead [1479]. SMC and AIS incur a per-sample overhead in cost ($\sum_t m_t$, where m_t are the mutations at tempering step t), but this is offset by intra-parallelism ($\sum_t m_t/M \ll 1$).

IMDb dataset

We now consider a large problem in semantic classification of natural language, using the IMDb dataset [1100] of 50,000 movie reviews, split evenly into training and test sets.

Architecture. Each review is mapped to a fixed feature vector by SBERT embeddings [1388] from the model `all-mpnet-base-v2` [1539]¹, whose weights are *frozen* throughout: they are used only to produce the 768 dimensional [CLS] embedding and are never sampled. On top of these embeddings we place (i) one hidden layer with 128 neurons, (ii) ReLU activation, (iii) a final linear layer, and (iv) a softmax output. Inference is over *all* weights and biases of this network, of dimension $P = 98690$, and not merely the final classifier head. The whole train (25000 data) and test dataset (25000 data) are considered.

Sampling setup. The likelihood is the softmax (categorical cross-entropy) likelihood over the whole training set of 25000 reviews, and inference uses an adaptive likelihood-tempering SMC sampler with HMC mutations of length T . We compare $\text{SMC}_{\parallel}$ against two baselines that share the same prior, likelihood, and frozen embeddings: $\text{HMC}_{\parallel}$, which averages over MR independent HMC chains of length T , and a single serial HMC run of length MRT ; in every method the HMC step size is adapted to a common acceptance-rate target, so the comparison is made at matched cost.

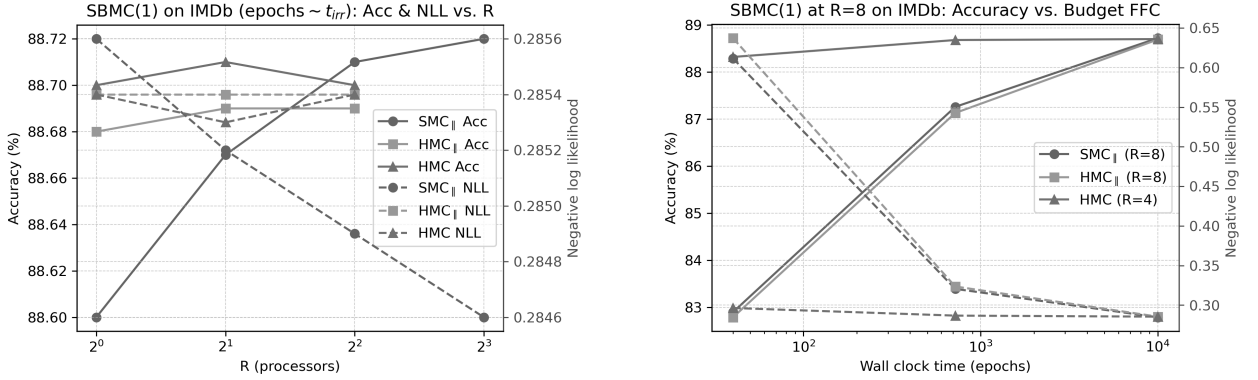


Figure 4.4: IMDb sentiment classification, with the full posterior over all $P = 98690$ network weights, using $\text{SMC}_{\parallel}$ with $M = 32$ particles for each of R SMC runs, HMC mutations, and adaptive tempering that keeps the effective sample size above $M/2$. Left: test accuracy and negative log-likelihood (NLL) against the number of parallel runs R at a budget of about $\tau_{\text{irr}} = 10^4$ epochs, alongside a serial HMC baseline given $MR\tau_{\text{irr}}$ epochs and $\text{HMC}_{\parallel}$ with MR chains with τ_{irr} epochs each. Right: converged values ($R = 8$) showing the catastrophic failure (collapsed accuracy and inflated NLL) once the budget drops below τ_{irr} . The R - and budget-axes are log-scaled; all curves are averaged over 5 realizations.

Experiment. Denote by m the number of evaluations of κ per mutation step, which we refer to as *epochs*, and by T the number of tempering-and-mutation steps. The product mT (or $\sum_{t=1}^T m_t$ for an adaptive schedule) is the total simulation cost, which we call the *irreducible serial time* τ_{irr} : the serial simulation the sampler needs in order to mix, which cannot be removed by adding more parallel runs. In practice τ_{irr} is read off empirically as the budget at which accuracy and NLL plateau, here $\tau_{\text{irr}} \approx 10^4$ epochs. When m and/or T are too small, so that the budget falls well short of τ_{irr} , the sampler has not mixed and the method *catastrophically fails*: test accuracy collapses and NLL inflates. Figure 4.4 summarises this behaviour. The left panel shows performance as a function of R at τ_{irr} , where $\text{SMC}_{\parallel}$ improves steadily with R and matches the far more expensive serial HMC baseline; the right panel fixes $R = 8$ and reduces the budget below τ_{irr} , exposing the catastrophic failure of $\text{SMC}_{\parallel}$ and $\text{HMC}_{\parallel}$ once the budget falls short of the irreducible time.

¹<https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

Chapter 5

Low-precision sampling

When running simulations for a dynamic system on a computer, the impact of round-off errors on number representations (e.g., floating point representations) can not be naïvely presumed to be negligible. It has been observed that a large deviation can occur between numerical behavior and the theoretical behaviour [684]. In Bayesian literature, the same question arises: how does the rounding error from computer simulation affect the convergence property of a Markov chain? Roberts et al. [1411] shows a rather alarming example that a well-behaved Markov chain (with Feller continuity and geometrical ergodicity) becomes transient (i.e., will not converge to any target distribution) after adding an arbitrarily small roundoff error. Hoffman [764] also warned that roundoff errors can severely degrade the performance of the Metropolis-Hastings sampling algorithm. To sum up, a common wisdom for conventional MCMC practice is to utilize high-precision numerical representations to limit the negative impact of roundoff errors. However, in this chapter, we will embrace the low-precision representations in the era of SG-MCMC samplers.

5.1 Low-Precision Deep Neural Network

The rise of large-scale deep neural networks (DNNs) and large language models (LLMs) has led to significant computational and memory costs of model training and inference. To improve efficiency, low-precision deep learning [800, 827, 1028, 1383, 1843, 1873] — which uses fewer bits to represent numbers in model parameters, activations, and gradients, and thus can drastically lower resource demands — has emerged as a promising direction. Precision formats such as 16-bit (FP16), 8-bit (INT8), 4-bit, and even binary (1-bit) offer memory reduction, energy savings, and speedup in inference, enabling deployment on edge devices and platforms, and specialized accelerators. On the other hand, well-designed low-precision optimization algorithms can maintain the model utility, such that the performance degradation from full-precision models to low-precision models is tolerable.

Modern foundations of low-precision deep learning include Gupta et al. [689], which leverages 16-bit wide fixed-point arithmetic and stochastic rounding to accelerate the training speed of DNNs, and Vanhoucke et al. [1666], which discusses the possibility of running DNN inference on x86 CPUs rather than GPUs using 8-bit fixed-point arithmetic. In general, there are two directions for achieving lower precision representations of DNN models: quantization-aware training (QAT) and post-training quantization (PTQ). The former simulates quantization during the training process, allowing the model to learn a robust solution that accounts for quantization errors induced by rounding; the latter, on the other hand, applies quantization to off-the-shelf pre-trained models. It is arguably true that QAT outperforms PTQ in terms of better accuracy, but incurs greater computational costs of model training. Other worth mentioning techniques of low-precision models are, but not limited to, non-uniform quantization and mixed precision training. Opposite to conventional quantization methods that map floating point values to an evenly spaced quantized target space, non-uniform quantization chooses an unevenly spaced quantized target space that can allocate more quantization levels where the values are dense. This unevenly spaced quantized target space can be of logarithmic scale [1191], or adaptively learned from the training process [1843]. Mixed-precision methods [440, 1171] assign different bit-widths to different layers of the neural networks or different computation steps of the training, leveraging the fact that different components of DNNs have varying sensitivities to quantization error. Theoretical investigations related to low-precision training are available in Li et al. [1028], Markov et al. [1134], Sa et al. [1445], Sakr et al. [1449].

Inevitably, low-precision optimization achieves better efficiency at the expense of performance sacrifice. One natural way to mitigate performance degradation caused by quantization error is low-precision Bayesian sampling. We argue that posterior sampling is particularly suited for low-precision arithmetic because of its intrinsic ro-

bustness to system noise. In particular: (1) Bayesian posterior explores weight space instead of converging to a single point, thus it should not require precise weights and be tolerant to non-full-precision models; and (2) it allows us to perform Bayesian model averaging during testing using an ensemble of models, which allows coarse representations of individual models to be compensated by the overall model average [1878].

5.2 Low-Precision SG-MCMC

5.2.1 Preliminary

The Bayesian counterpart of QAT directly samples low-precision models (i.e., low-precision weights of the neural network) from a target distribution (typically, a Bayesian posterior distribution). A straightforward idea is to adapt existing Monte Carlo sampling algorithms for low-precision sampling tasks. In the regime of deep learning, where the training data sets are usually massive, the SG-MCMC sampling algorithm (see Chapter 2) is the most suitable choice since it can avoid a full scan of the data set when calculating the likelihood terms.

Now we briefly recall two seminal SG-MCMC algorithms, the SGLD and SGHMC. Given some dataset $\mathcal{D}$, a model with parameters $\theta \in \mathbb{R}^P$, and a prior $p(\theta)$, we are interested in sampling from the posterior $p(\theta|\mathcal{D}) \propto \exp(-U(\theta))$, where the energy function is

$$U(\theta) = - \sum_{(x,y) \in \mathcal{D}} \log p(x, y|\theta) - \log p(\theta).$$

Stochastic Gradient Langevin Dynamics (SGLD) reduces the computational cost by using a stochastic gradient estimation $\nabla \tilde{U}$, which is an unbiased estimator of ∇U based on a subset of the dataset $\mathcal{D}$. Specifically, SGLD updates the parameter in the $(k+1)$ -th step following the rule

$$\theta_{k+1} = \theta_k - \alpha \nabla \tilde{U}(\theta_k) + \sqrt{2\alpha} \xi_{k+1}, \quad (5.1)$$

where α is the stepsize and ξ_{k+1} is a standard Gaussian noise. Compared to the Stochastic Gradient Descent (SGD) update, the only difference is that SGLD adds an additional Gaussian noise in each step, which essentially enables SGLD to characterize the full distribution instead of converging to a single point. The close connection between SGLD and SGD makes it convenient to implement and run on existing deep learning tasks for which SGD is the typical learning algorithm.

Stochastic Gradient HMC [SGHMC, 273] is derived by replacing ∇U with a noisy gradient $\nabla \tilde{U} + \xi$ in the Hamiltonian Monte Carlo, and boils down to the unadjusted underdamped Langevin algorithm with stochastic gradient, where the underdamped Langevin dynamics can be defined as

$$\begin{aligned} dv_t &= -\gamma v_t dt - u \nabla U(\theta_t) dt + \sqrt{2\gamma u} d\mathbf{B}_t \\ d\theta_t &= v_t dt, \end{aligned} \quad (5.2)$$

where u, γ denote the hyperparameters of the inverse mass and friction, respectively. Furthermore, Cheng et al. [287] proposed the following discretization of underdamped Langevin dynamics (5.2) with stochastic gradient:

$$\begin{aligned} v_{k+1} &= v_k e^{-\gamma\eta} - u\gamma^{-1}(1 - e^{-\gamma\eta})\nabla \tilde{U}(\theta_k) + \xi_k^v \\ \theta_{k+1} &= \theta_k + \gamma^{-1}(1 - e^{-\gamma\eta})v_k + u\gamma^{-2}(\gamma\eta + e^{-\gamma\eta} - 1)\nabla \tilde{U}(\theta_k) + \xi_k^\theta, \end{aligned} \quad (5.3)$$

where ξ_k^v, ξ_k^θ are normal distributed in $\mathbb{R}^P$ satisfying that :

$$\begin{aligned} \mathbb{E}\xi_k^v(\xi_k^v)^\top &= u(1 - e^{-2\gamma\eta}) \cdot \mathbf{I}, \\ \mathbb{E}\xi_k^\theta(\xi_k^\theta)^\top &= u\gamma^{-2}(2\gamma\eta + 4e^{-\gamma\eta} - e^{-2\gamma\eta} - 3) \cdot \mathbf{I}, \\ \mathbb{E}\xi_k^\theta(\xi_k^v)^\top &= u\gamma^{-1}(1 - 2e^{-\gamma\eta} + e^{-2\gamma\eta}) \cdot \mathbf{I}. \end{aligned} \quad (5.4)$$

Before formally introducing the low-precision SG-MCMC algorithms, we review the basic concepts of low-precision training. To represent numbers, e.g., the weights of a DNN, in low-precision, one simple way is to use *fixed point*, which has been utilized in both theory and practice [689, 1028, 1797]. Specifically, suppose that we use W bits to represent numbers with F of those W bits to represent the fractional part. Then there is a distance between consecutive representable numbers, $\Delta = 2^{-F}$, which is called *quantization gap*. The representable numbers also have a lower bound l and an upper bound u , where

$$L = -2^{W-F-1}, \quad U = 2^{W-F-1} - 2^{-F}.$$

As shown above, when the number of bits decreases, the accuracy of number representation decreases. We use this type of number representation in our theoretical analysis and empirical demonstration following previous work [1028, 1797].

Another common type of number representation is *floating point*, where each number has its own exponent part. Between fixed point and floating point, there is *block floating point* which allows all numbers within a block to share the same exponent [1543]. We use block floating point for deep learning experiments since it has been shown to be more favourable for deep models [1797].

Having low-precision number representation in hand, we also need a quantization function Q to convert a real-valued number into a low-precision number. Such functions include *deterministic rounding* and *stochastic rounding*. Particularly, the deterministic rounding function Q^d quantizes a number to its nearest representable neighbour as follows:

$$Q^d(\theta) = \text{sign}(\theta) \cdot \text{clip} \left(\Delta \left\lfloor \frac{|\theta|}{\Delta} + \frac{1}{2} \right\rfloor, L, U \right),$$

where $\text{clip}(x, L, U) = \max(\min(x, U), L)$ and the above action is applied to the whole vector elementwisely. Instead, stochastic rounding Q^s quantizes a number with a probability based on the distance to its representable neighbour:

$$Q^s(\theta) = \begin{cases} \text{clip} \left(\Delta \left\lfloor \frac{\theta}{\Delta} \right\rfloor, L, U \right), & \text{w.p. } \left\lceil \frac{\theta}{\Delta} \right\rceil - \frac{\theta}{\Delta} \\ \text{clip} \left(\Delta \left\lceil \frac{\theta}{\Delta} \right\rceil, L, U \right), & \text{w.p. } 1 - \left(\left\lceil \frac{\theta}{\Delta} \right\rceil - \frac{\theta}{\Delta} \right). \end{cases}$$

An important property of Q^s is that $\mathbb{E}[Q^s(\theta)] = \theta$, which means the quantized number is unbiased. Q^s is generally preferred over Q^d in practice since it can preserve gradient information, especially when the gradient update is smaller than the quantization gap [689, 1708]. In what follows, when necessary, we use Q_W and Δ_W to denote the weights' quantizer and quantization gap, Q_G and Δ_G to denote gradients' quantizer and quantization gap.

To do the gradient update in low-precision training optimization, there are two common choices depending on whether we store an additional copy of full-precision weights. *Full-precision gradient accumulators* use a full-precision weight buffer to accumulate gradient updates and only quantize weights before computing gradients. SGD with full-precision gradient accumulators (SGDLP-F) updates the weights as follows,

$$\theta_{k+1} = \theta_k - \alpha Q_G \left(\nabla \tilde{U}(Q_W(\theta_k)) \right),$$

where we use full-precision θ_{k+1} and θ_k in the update, and only quantize the weight for forward and backward propagation [334, 1028].

However, gradient accumulators have to be frequently updated during training. It would be ideal to also represent it in low-precision to further reduce the costs. To achieve it, we could instead do the update as follows,

$$\theta_{k+1} = Q_W \left(\theta_k - \alpha Q_G \left(\nabla \tilde{U}(\theta_k) \right) \right), \quad (5.5)$$

where θ is always represented in low-precision. This update of SGD is called using *low-precision gradient accumulators* (SGDLP-L). Both full- and low-precision gradient accumulators have been widely used: low-precision gradient accumulators are cheaper and faster because of having all numbers in low-precision, whereas full-precision gradient accumulators generally have better performance because of more precisely reflecting small gradient updates [334, 1028].

5.2.2 Low-Precision Stochastic Gradient Langevin Dynamics

In this section, we combine the low-precision rounding and the SGLD update (5.1). As shown in Equation (5.1), the update of SGLD is simply an SGD update plus a Gaussian noise. Therefore, the low-precision formulation for SGD in the previous section can be naturally extended to SGLD training. Similar to SGDLP-F, we can do low-precision SGLD with full-precision gradient accumulators (SGLDLP-F) as follows:

$$\theta_{k+1} = \theta_k - \alpha Q_G \left(\nabla \tilde{U}(Q_W(\theta_k)) \right) + \sqrt{2\alpha} \xi_{k+1}, \quad (5.6)$$

which can also be viewed as SGDLP-F plus a Gaussian noise in each step. However, the Gaussian noise turns out to help counteract the noise introduced by quantization, making SGLDLP-F more robust to inaccurate number representation and converging better than SGDLP-F as shown in Theorem 5.1.

To facilitate the theoretical analysis, we assume that the target distribution is smooth and strongly log-concave, and the energy function has Lipschitz Hessian, i.e., $\forall \theta, \theta' \in \mathbb{R}^P$,

$$\begin{aligned} U(\theta) - U(\theta') - \nabla U(\theta')^\top (\theta - \theta') &\geq (m/2) \|\theta - \theta'\|_2^2, \\ \|\nabla U(\theta) - \nabla U(\theta')\|_2 &\leq M \|\theta - \theta'\|_2, \\ \|\nabla^2 U(\theta) - \nabla^2 U(\theta')\|_2 &\leq \Psi \|\theta - \theta'\|_2, \\ \mathbb{E}[\|\nabla \tilde{U}(\theta) - \nabla U(\theta)\|_2^2] &\leq \kappa^2. \end{aligned} \quad (5.7)$$

These assumptions are commonly used in the literature [e.g., 359, 1719, 1797, 1856].

Theorem 5.1. *We run SGLDLP-F under the above assumptions and with a constant stepsize $\alpha \leq 2/(m + M)$. Let π be the target distribution, μ_0 be the initial distribution and μ_K be the distribution obtained by SGLDLP-F after K iterations, then the 2-Wasserstein distance is*

$$\begin{aligned} W_2(\mu_K, \pi) &\leq (1 - \alpha m)^K W_2(\mu_0, \pi) + 1.65(M/m)(\alpha P)^{1/2} \\ &\quad + \min\left(\frac{\Psi \Delta_W^2 P}{4m}, \frac{M \Delta_W \sqrt{P}}{2m}\right) + \sqrt{\frac{(\Delta_G^2 + M^2 \Delta_W^2) \alpha P + 4\alpha \kappa^2}{4m}}. \end{aligned}$$

This theorem shows that SGLDLP-F converges to the accuracy floor at

$$\min\left(\frac{\Psi \Delta_W^2 P}{4m}, \frac{M \Delta_W \sqrt{P}}{2m}\right) \quad (5.8)$$

given a large iteration number K and small stepsize α . Besides, if we further assume that the energy function is quadratic (i.e., $\Psi = 0$), then SGLDLP-F converges to the target distribution asymptotically. This matches the result of SGDLP-F on a quadratic function which converges to the optimum asymptotically [1028]. Our theorem also recovers the bound of full-precision SGLD (i.e., $\Delta_W = 0$) in Dalalyan and Karagulyan [359]. However, when the energy function is not quadratic, the convergence of SGLDLP-F to the target distribution has a $\mathcal{O}(\Delta_W^2)$ rate whereas SGDLP-F to the optimum has a $\mathcal{O}(\Delta_W)$ rate [1797]. Recall that $\Delta_W = 2^{-F}$ where F is the number of fractional bits. Our result suggests that asymptotically, SGLD only needs half the number of bits as SGD needs to achieve the same convergence accuracy! This comparison between SGLD and SGD also fits into the literature in comparing sampling and optimization convergence bounds [1098, 1586].

To further reduce the costs using low-precision gradient accumulators beyond SGLDLP-F, we mimic the update of SGDLP-L in Equation (5.5) and naturally get the following update rule for SGLD with low-precision gradient accumulators (SGLDLP-L),

$$\theta_{k+1} = Q_W \left(\theta_k - \alpha Q_G \left(\nabla \tilde{U}(\theta_k) \right) + \sqrt{2\alpha} \xi_{k+1} \right). \quad (5.9)$$

Surprisingly, despite the convergence of SGLDLP-L, the next theorem shows that it may diverge arbitrarily far away from the target distribution with small stepsizes.

Theorem 5.2. *We run SGLDLP-L under the same assumptions as in Theorem 5.1. Let μ_0 be the initial distribution and μ_K be the distribution obtained by SGLDLP-L after K iterations, then*

$$\begin{aligned} W_2(\mu_K, \pi) &\leq (1 - \alpha m)^K W_2(\mu_0, \pi) + 1.65(M/m)(\alpha P)^{1/2} + \min\left(\frac{\Psi \Delta_W^2 P}{4m}, \frac{M \Delta_W \sqrt{P}}{2m}\right) \\ &\quad + \sqrt{\frac{(\alpha \Delta_G^2 + \alpha^{-1} \Delta_W^2) P + 4\alpha \kappa^2}{4m}} + ((1 - \alpha m)^K + 1) \frac{\Delta_W \sqrt{P}}{2}. \end{aligned}$$

Note that one term on the right-hand side contains α^{-1} , this theorem implies that as the stepsize α decreases, W_2 distance between the stationary distribution of SGLDLP-L and the target distribution may increase. To see if this is the case, we empirically run SGLDLP-L on a standard Gaussian distribution in Figure 5.1. We use 8-bit fixed-point and assign 3 of them to represent the fractional part. The simulations verify that SGLDLP-L indeed diverges from the target distribution with small stepsizes. At the same time, SGLDLP-F always converges to the target distribution with different stepsizes, aligning with the result in Theorem 5.1.

One may choose a stepsize that minimizes the above W_2 distance to avoid divergence, however, getting that optimal stepsize is, in general, difficult since the constants are unknown in practice. Moreover, enabling a small

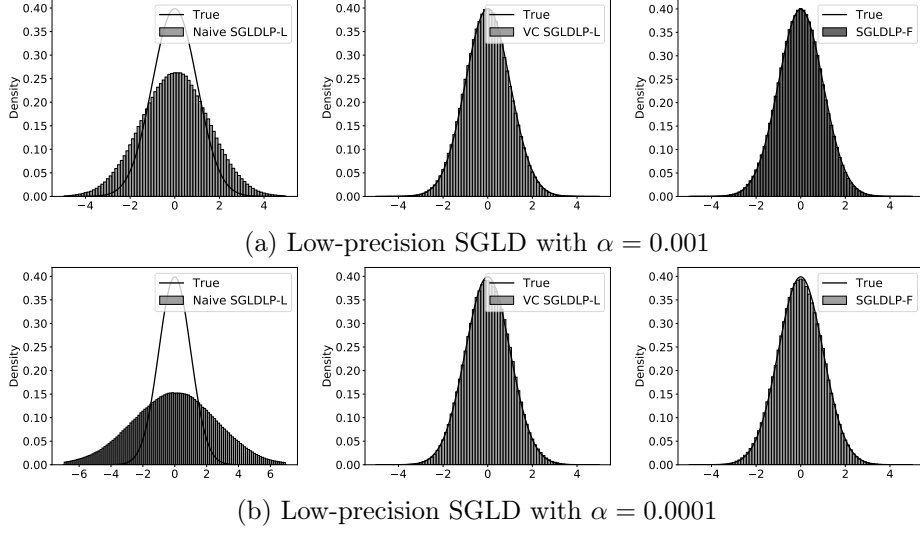


Figure 5.1: Low-precision SGLD with varying stepsizes on a Gaussian distribution. Variance-corrected SGLD with low-precision gradient accumulators (VC SGLDLP-L) and SGLD with full-precision gradient accumulators (SGLDLP-F) converge to the true distribution, whereas naïve SGLDLP-L diverges and the divergence increases as the stepsize decreases.

stepsize in SGLD is often desirable, since it is needed to reduce the asymptotic bias of the posterior approximation [1737].

One key reason causing the divergence of SGLDLP-L is that the *variance* of each dimension of θ_{k+1} becomes larger due to using low-precision gradient accumulators. Specifically, given the stochastic gradient $\nabla \tilde{U}$, the update of full-precision SGLD can be viewed as sampling from a Gaussian distribution for each dimension i

$$\theta_{k+1,i} \sim \mathcal{N}\left(\theta_{k,i} - \alpha \nabla \tilde{U}(\theta_k)_i, 2\alpha\right), \text{ for } i = 1, \dots, P.$$

Since stochastic rounding is unbiased, using it as the weight quantizer Q_W and the gradient quantizer Q_G in SGLDLP-L gives us

$$\mathbb{E}\theta_{k+1,i} = \mathbb{E}Q_W\left(\theta_{k,i} - \alpha Q_G\left(\nabla \tilde{U}(\theta_k)\right)_i + \sqrt{2\alpha}\xi_{k+1,i}\right) = \theta_{k,i} - \alpha \nabla \tilde{U}(\theta_k)_i,$$

which has the same mean as θ_{k+1} in full-precision. However, the variance of $\theta_{k+1,i}$ is now essentially larger than needed. If we ignore the variance from Q_G and the stochastic gradient, since they are present and have been shown to work well in SGLDLP-F, the variance of $\theta_{k+1,i}$ is

$$\begin{aligned} \text{Var}(\theta_{k+1,i}) &= \mathbb{E}\left[\text{Var}\left[Q_W\left(\theta_{k,i} - \alpha \nabla U(\theta_k)_i + \sqrt{2\alpha}\xi_{k+1,i}\right) \middle| \xi_{k+1,i}\right]\right] \\ &\quad + \text{Var}\left[\mathbb{E}\left[Q_W\left(\theta_{k,i} - \alpha \nabla U(\theta_k)_i + \sqrt{2\alpha}\xi_{k+1,i}\right) \middle| \xi_{k+1,i}\right]\right] \\ &= \frac{\Delta_W^2}{4}\chi_{k+1,i} + 2\alpha, \end{aligned}$$

where $\chi_{k+1,i} \in [0, 1]$ depends on the distance of $\theta_{k+1,i}$ to its discrete neighbour. The above equation shows that the variance in SGLDLP-L update is larger than the right variance value 2α . This can be validated from Figure 5.1 where the naïve SGLDLP-L estimates the mean correctly but the variance wrongly.

To enable SGLD with low-precision gradient accumulators, we introduce a new quantization function Q^{vc} , denoting variance-corrected quantization, to fix the issue. The main idea of Q^{vc} is to directly sample from the discrete weight space instead of quantizing a real-valued Gaussian sample. First, we note that if we want a sample with mean $\mu \geq 0$ and variance $v \leq \Delta_W^2/4$, we could sample from the following categorical distribution over

$\{\Delta_W, -\Delta_W, 0\}$ to get it,

$$\text{Cat}(\mu, v) = \begin{cases} \Delta_W, & w.p. \frac{v+\mu^2+\mu\Delta_W}{2\Delta_W^2} \\ -\Delta_W, & w.p. \frac{v+\mu^2-\mu\Delta_W}{2\Delta_W^2} \\ 0, & \text{otherwise} \end{cases} \quad (5.10)$$

providing that the probability values are valid (i.e., non-negative and smaller than 1). Now we show how to use this categorical distribution to preserve the correct mean and variance for quantized θ_{k+1} . We do so considering two cases: when the Gaussian variance 2α is larger than the largest possible stochastic rounding variance $\Delta_W^2/4$, Q^{vc} first adds a small Gaussian noise and uses a sample from Equation (5.10) to make up the remaining variance; in the other situation, Q^{vc} directly samples from Equation (5.10) to achieve the target variance. The full description of Q^{vc} is outlined in Algorithm 13 (where the use of $\text{Cat}(\cdot, \cdot)$ sampler is valid). SGLDLP-L using Q^{vc} , denoting VC SGLDLP-L, follows the update rule:

$$\theta_{k+1} = Q^{\text{vc}} \left(\theta_k - \alpha Q_G \left(\nabla \tilde{U}(\theta_k) \right), 2\alpha, \Delta_W \right). \quad (5.11)$$

Algorithm 13: Variance-Corrected Quantization Function Q^{vc} .

Input : (μ, v, Δ)

Output: $\theta \in \mathbb{R}^P$ with mean vector $\mu \in \mathbb{R}^P$ and coordinate variance $v \in \mathbb{R}$

$v_0 \leftarrow \Delta^2/4$ // $\Delta^2/4$ is the largest variance due to stochastic rounding

if $v > v_0$ **then**

$x \leftarrow \mu + \sqrt{v - v_0} \xi$, where $\xi \sim \mathcal{N}(0, I_P)$

$r \leftarrow x - Q^d(x)$

foreach i **do**

sample c_i from $\text{Cat}(|r_i|, v_0)$ as in (5.10)

 // c_i , r_i and θ_i are i th coordinates of c , r and θ , respectively

$\theta \leftarrow Q^d(x) + \text{sign}(r) \odot c$

else

$r \leftarrow \mu - Q^s(\mu)$

foreach i **do**

$v_s \leftarrow \left(1 - \frac{|r_i|}{\Delta}\right) r_i^2 + \frac{|r_i|}{\Delta} (-r_i + \text{sign}(r_i)\Delta)^2$

if $v > v_s$ **then**

sample c_i from $\text{Cat}(0, v - v_s)$ as in (5.10)

$\theta_i \leftarrow Q^s(\mu)_i + c_i$

else

$\theta_i \leftarrow Q^s(\mu)_i$

clip θ if outside representable range

return θ

Our variance-corrected quantization function Q^{vc} always guarantees the correct mean, $\mathbb{E}(\theta_{k+1,i}) = \theta_{k,i} - \alpha \nabla \tilde{U}(\theta_k)_i$, and further guarantees the correct variance $\text{Var}(\theta_{k+1,i}) = 2\alpha$ most of the time except when $v = 2\alpha < v_s$, where v_s is defined in Algorithm 13. However, that case rarely happens in practice, because the stepsize has to be extremely small. Besides, our quantization is simple to implement, and its computation cost is negligible compared to gradient computation. Although Q^{vc} only preserves the correctness of the first two moments (i.e., mean and variance), this does not affect the performance much in both theory and practice, as shown by Theorem 5.3 and later numerical evidence.

Theorem 5.3. *We run VC SGLDLP-L as in (5.11). Besides the same assumptions in Theorem 5.1, we further assume the gradient is bounded $\mathbb{E} \left[\left\| Q_G(\nabla \tilde{U}(\theta_k)) \right\|_1 \right] \leq G$. Let $v_0 = \Delta_W^2/4$. Then*

$$\begin{aligned} W_2(\mu_K, \pi) &\leq (1 - \alpha m)^K W_2(\mu_0, \pi) + 1.65(M/m)(\alpha P)^{1/2} \\ &\quad + \min \left(\frac{\Psi A}{m}, \frac{M\sqrt{A}}{m} \right) + \sqrt{\frac{\alpha \Delta_G^2 P + 4\alpha \kappa^2}{4m} + \frac{A}{\alpha m}} + ((1 - \alpha m)^K + 1) \sqrt{A}, \end{aligned}$$

	Test Errors (%)		ECE ↓(%)	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
32-bit Floating Point				
SGLDFP	4.65 ±0.06	22.58 ±0.18	1.11	3.92
SGDFP	4.71 ±0.02	22.64 ±0.13	2.53	4.97
8-bit Fixed Point				
VC SGLDLP-L	7.13 ±0.01	26.62 ±0.16	0.60	3.19
SGDLP-L	8.53 ±0.08	28.86 ±0.10	3.40	10.38
SGLDLP-F	5.12±0.06	23.30 ±0.09	1.12	4.42
SGDLP-F	5.20 ±0.14	23.84 ±0.12	3.05	6.80
8-bit Floating Point				
VC SGLDLP-L	5.51 ±0.01	25.22 ±0.18	0.60	5.82
SGDLP-L	5.86 ±0.18	26.19 ±0.11	4.23	12.97
SGLDLP-F	4.58 ±0.07	22.59 ±0.18	1.19	3.78
SGDLP-F	4.75 ±0.05	22.90 ±0.13	2.76	5.20

Table 5.1: Test Error and ECE on CIFAR dataset with the ResNet-18 model.

$$\text{where } A = \begin{cases} 5v_0P, & \text{if } 2\alpha > v_0 \\ \max(2\Delta_W\alpha G, 4\alpha P), & \text{otherwise} \end{cases}$$

This theorem shows that when the stepsize $\alpha \rightarrow 0$, VC SGLDLP-L converges to the target distribution up to an error instead of diverging. Moreover, VC SGLDLP-L converges to the target distribution in $\mathcal{O}(\sqrt{\Delta_W})$, which is equivalent to the convergence rate of SGD with low-precision gradient accumulators to the optimum [1028, 1797].

Numerical Experiments We demonstrate the generalization accuracy and uncertainty estimation of low-precision SGLD with full-precision gradient accumulators (SGLDLP-F) and with variance-corrected low-precision gradient accumulators (VC SGLDLP-L) on the ResNet-18 model on the CIFAR datasets [933, 934]. We use the same quantization for weights, gradients, activations, and backpropagation errors unless otherwise stated. For all experiments, SGLD collects samples from the posterior of the model’s weight and obtains the prediction on test data by Bayesian model averaging. We mainly compare low-precision SGLD with low-precision SGD, which is commonly used for frequentist low-precision deep learning [1577, 1578]. We use SGDFP, SGDLP-F, and SGDLP-L to denote full-precision SGD, low-precision SGD with full-precision gradient accumulators, and low-precision SGD with low-precision gradient accumulators, respectively. The numerical results are provided in Table 5.1, which reports average performance over 3 runs with the standard error.

1. Fixed Point: We use 8-bit fixed point for weights and gradients but full-precision for activations since we find low-precision activations significantly harm the performance. We find that, SGLDLP-F is better than SGDLP-F and VC SGLDLP-L significantly outperforms SGDLP-L across datasets and architectures. Notably, the improvement of SGLD over SGD becomes larger when using more low-precision arithmetic. For example, on CIFAR-100, VC SGLDLP-L outperforms SGDLP-L by 2.24%, SGLDLP-F outperforms SGDLP-F by 0.54% and SGLDFP outperforms SGDFP by 0.06%. This demonstrates that SGLD is particularly compatible with low-precision deep learning because of its natural ability to handle system noise.
2. Block Floating Point: We also consider block floating point (BFP) which is another common number type and is often preferred over fixed point on deep models due to less quantization error caused by overflow and underflow [1543]. Following the block design in Yang et al. [1797], we use *small-block* for ResNet. The Q^{vc} function naturally generalizes to BFP and only needs a small modification (see Zhang et al. [1856] for the detailed algorithm of Q^{vc} with BFP). By using BFP, the results of all low-precision methods improve over fixed point. SGLDLP-F can match the performance of SGLDFP with all numbers quantized to 8-bit except gradient accumulators. Again, SGLDFP-F and VC SGLDLP-L outperform their SGD counterparts on all tasks, suggesting the general applicability of low-precision SGLD with different number types.
3. Expected Calibration Error: Besides generalization performance, we further report the results of expected calibration error (ECE) [687] to demonstrate the uncertainty estimation of low-precision SGLD. In Table 5.1, we observe that SGLDLP-F and VC SGLDLP-L achieve almost the same or even lower ECE than full-precision SGLD, showing the ability of SGLD to give well-calibrated predictions does not degenerate due

to using low-precision. VC SGLDLP-L sometimes gives lower ECE than SGLDLP-F which may be due to the regularization effect of low-precision arithmetic. In contrast, the ECE of low-precision SGD increases significantly compared to full-precision SGD, implying that the quantization error makes the standard DNNs even more overconfident, which might lead to wrong decisions in real-world applications.

5.2.3 Low-Precision Stochastic Gradient Hamiltonian Monte Carlo

In this section, we combine low-precision arithmetic with SGHMC algorithm (5.3). Recall that SGHMC is equivalent to a discretized underdamped Langevin algorithm with stochastic gradient. The stationary distribution of underdamped Langevin dynamics is defined on the augmented probability space: $\pi(\theta, v) \propto \exp(-U(\theta) - \|v\|_2^2/2u)$. Therefore, our theoretical investigation will investigate the 2-Wasserstein distance¹ between the distribution of (θ_K, v_K) and $\pi(\theta, v)$.

Adopting the update rule in equations (5.3), we introduce low-precision SGHMC with full gradient accumulators (SGHMCLP-F) as follows:

$$\begin{aligned} v_{k+1} &= v_k e^{-\gamma\eta} - u\gamma^{-1}(1 - e^{-\gamma\eta})Q_G(\nabla\tilde{U}(Q_W(\theta_k))) + \xi_k^v \\ \theta_{k+1} &= \theta_k + \gamma^{-1}(1 - e^{-\gamma\eta})v_k + u\gamma^{-2}(\gamma\eta + e^{-\gamma\eta} - 1)Q_G(\nabla\tilde{U}(Q_W(\theta_k))) + \xi_k^\theta, \end{aligned} \quad (5.12)$$

where the distributions of ξ^v and ξ^θ follow (5.4). SGHMCLP-F keeps full-precision parameters v_k, θ_k at each iteration and quantizes them to low-precision representations before taking the gradient. Following the previous section, we still assume strong convexity of $U(\theta)$. In addition, we denote $\theta^* = \arg \min_\theta U(\theta)$ and $\rho = M/m$, and assume $\Delta_G = \Delta_W = \Delta$ for the simplicity of analysis presented in the following sections.

Theorem 5.4. *Under assumptions of Theorem 5.1, we assume that the minimum of $U(\theta)$ satisfies $\|\theta^*\|^2 < \mathcal{D}^2$. Given any sufficiently small $\epsilon > 0$, if we set the step size to be*

$$\eta = \min \left\{ \frac{\epsilon\rho^{-1}}{\sqrt{479232/5(P/m + \mathcal{D}^2)}}, \frac{\epsilon^2}{1440\rho u^2[(M^2 + 1)\frac{\Delta^2 P}{4} + \kappa^2]} \right\},$$

then after K steps starting with initial points $\theta_0 = v_0 = 0$, the distribution (θ_K, v_K) of the SGHMCLP-F (5.12), denoted by μ_K , satisfies

$$W_2(\mu_K, \pi) \leq \tilde{O}(\epsilon + \Delta),$$

for some K satisfying

$$K \leq \frac{\rho}{\eta} \log \left(\frac{36(\frac{P}{m} + \mathcal{D}^2)}{\epsilon} \right) = \tilde{O}(\epsilon^{-2} \log(\epsilon^{-1})(\Delta^2 + 1)),$$

where the big- O notation $\tilde{O}$ gives explicit dependence on the quantization error Δ but hides multiplicative terms that polynomially depend on the other parameters (e.g., dimension P , friction γ , inverse mass u , and gradient variance κ^2).

Theorem 5.1 implies that for a strongly log-concave target distribution, the low-precision SGLDLP-F also achieves ϵ accuracy within $\tilde{O}(\epsilon^{-2} \log(\epsilon^{-1})(\Delta^2 + 1))$ iterations with learning rate of order $\epsilon^2/(\Delta^2 + 1)$. Thus, under strong convexity, SGHMCLP-F achieves no better convergence speed than SGLDLP-F. This is not surprising, since the quantization applied to the gradients in the full-precision gradient accumulator algorithm is equivalent to adding extra noise to the stochastic gradients. As theoretically shown by Cheng et al. [287] for a strongly-log-concave target distribution, SGHMC doesn't exhibit any advantage over the overdamped Langevin algorithm when stochastic gradients are used.

To reduce the computation cost, the SGHMC with low-precision gradient accumulators (SGHMCLP-L) can be trivially implemented as

$$\begin{aligned} v_{k+1} &= Q_W \left(v_k e^{-\gamma\eta} - u\gamma^{-1}(1 - e^{-\gamma\eta})Q_G(\nabla\tilde{U}(\theta_k)) + \xi_k^v \right), \\ \theta_{k+1} &= Q_W \left(\theta_k + \gamma^{-1}(1 - e^{-\gamma\eta})v_k + u\gamma^{-2}(\gamma\eta + e^{-\gamma\eta} - 1)Q_G(\nabla\tilde{U}(\theta_k)) + \xi_k^\theta \right). \end{aligned} \quad (5.13)$$

¹The 2-Wasserstein distance between two probability measures μ and ν is defined as $W_2(\mu, \nu) = \left(\int_{\Gamma(\mu, \nu)} \|x - y\|^2 d\gamma(x, y) \right)^{1/2}$, where $\Gamma(\mu, \nu)$ denotes the set of joint distributions whose marginals are exactly μ and ν .

Similar to SGLDLP-L, SGHMCLP-L has intrinsic flaws. It directly quantizes the weights after each update, hence a small stepsize update is often quantized to zero update of θ and v , resulting in the sample distribution converging to a Dirac distribution at the initial point. In such cases, ensuring convergence becomes challenging. Wang et al. [1719] investigated the convergence of SGHMCLP-L under 2-Wasserstein distance, exhibiting that the naïve SGHMCLP-L has a much worse convergence upper bound than SGHMCLP-F. To remedy this, we adapt the variance correction quantizer (Algorithm 13) to SGHMCLP-L. Let $\mathbb{V}_v^{hmc} = u(1 - e^{-2\gamma\eta})$ and $\mathbb{V}_\theta^{hmc} = u\gamma^{-2}(2\gamma\eta + 4e^{-\gamma\eta} - e^{-2\gamma\eta} - 3)$, which are the desired variances induced from the underdamped Langevin dynamics (5.3). The VC SGHMCLP-L can be done as follows:

$$\begin{aligned} v_{k+1} &= Q^{vc} \left(v_k e^{-\gamma\eta} - u\gamma^{-1}(1 - e^{-\gamma\eta})Q_G(\nabla\tilde{U}(\theta_k)), \mathbb{V}_v^{hmc}, \Delta \right), \\ \theta_{k+1} &= Q^{vc} \left(\theta_k + \gamma^{-1}(1 - e^{-\gamma\eta})v_k + u\gamma^{-2}(\gamma\eta + e^{-\gamma\eta} - 1)Q_G(\nabla\tilde{U}(\theta_k)), \mathbb{V}_\theta^{hmc}, \Delta \right). \end{aligned} \quad (5.14)$$

Next, we present the convergence result for VC SGHMCLP-L under a strongly log-concave target distribution.

Theorem 5.5. *Under assumptions of Theorem 5.1, we assume that the minimum of $U(\theta)$ satisfies $\|\theta^*\|^2 < \mathcal{D}^2$. Given any sufficiently small $\epsilon > 0$, if we set the stepsize to be*

$$\eta = \min \left\{ \frac{\epsilon}{479232/5 \left(\frac{P}{m} + \mathcal{D}^2 \right) \rho}, \frac{\epsilon^2}{90u^2\Delta^2 P\rho + 360u^2\kappa^2\rho} \right\}$$

after K steps starting from the initial point $\theta_0 = v_0 = 0$, the distribution (θ_K, v_K) of the VC SGHMCLP-L (5.14) satisfies

$$W_2(\mu_K, \pi) = \tilde{O} \left(\epsilon + \sqrt{\Delta} \right), \quad (5.15)$$

for some K satisfied

$$K \leq \frac{\rho}{\eta} \log \left\{ \frac{36 \left(\frac{P}{m} + \mathcal{D}^2 \right)}{\epsilon} \right\} = \tilde{O} \left(\epsilon^{-2} \log(\epsilon^{-1})(\Delta^2 + 1) \right).$$

Theorem 5.5 shows that the variance corrected quantization function resolves the overdispersion problem for the naïve SGHMCLP-L algorithm. The W_2 distance between the sample distribution and target distribution can be arbitrarily close to $\tilde{O}(\sqrt{\Delta})$. This convergence rate of VC SGHMCLP-L is the same as VC SGLDLP-L (see the discussion after Theorem 5.3).

5.2.4 Advantage of LP Stochastic Gradient HMC under Non-Convexity

So far, we have introduced the low-precision SGLD and SGHMC algorithms and explored their theoretical convergence behaviour. The theorems in previous sections don't showcase any advantage of low-precision SGHMC over low-precision SGLDLP for strongly log-concave distributions. This doesn't align with the common understanding of SGHMC performance. Cheng et al. [287], Gao et al. [575] showed that SGHMC converges to its target distribution faster than the best-known convergence rate of SGLD in the 2-Wasserstein distance under both strongly log-concave and non-log-concave assumptions. Beyond this, SGHMC is analogous to stochastic gradient methods augmented with momentum, which is shown to have more robust updates w.r.t. gradient estimation noise [1069]. Since the quantization-induced stochastic error in low-precision updates acts as extra gradient noise, we expect that low-precision SGHMC can perform better than low-precision SGLD under certain scenarios beyond strongly convexity of $U(\theta)$.

Wang et al. [1719] explored the convergence of low-precision SGHMC and SGLD under non-log-concave distributions. Instead of assuming $U(\theta) - U(\theta') - \nabla U(\theta')^\top(\theta - \theta') \geq (m/2)\|\theta - \theta'\|_2^2$, the *dissipative* condition is considered:

$$\langle \nabla U(\theta), \theta \rangle \geq m\|\theta\|^2 - b, \quad \text{for any } \theta \in \mathbb{R}^P.$$

Such an assumption is commonly used [575, 1362, 1886] in the analysis of sampling from non-log-concave distributions. The theorem below summarizes the theoretical findings of Wang et al. [1719], which emphasize how the convergence rate depends on the quantization gap Δ .

	Test Errors (%)		ECE ↓(%)	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
32-bit Floating Point				
SGHMC FP	4.78 ±0.08	22.37 ±0.04	0.72	1.52
8-bit Fixed Point				
VC SGHMCLP-L	6.60 ±0.06	26.43 ±0.19	0.70	2.44
SGHMCLP-F	5.08±0.08	23.54 ±0.10	1.11	1.92
8-bit Floating Point				
VC SGHMCLP-L	5.15 ±0.08	24.45 ±0.16	0.67	5.02
SGHMCLP-F	4.93 ±0.09	22.39 ±0.11	1.12	3.65

Table 5.2: Test Error and ECE on CIFAR dataset with the ResNet-18 model.

Theorem 5.6. *Under assumptions (5.7) except for the strongly convexity and dissipative assumption, there exist respective choices of tuning parameter (e.g., stepsize) and the number of iterations K , such that one can achieve*

$$\begin{aligned}
W_2(\mu_K, \pi) &= \tilde{O}\left(\Delta^{e/(1+2e)}\right), \text{ for SGHMCLP-F,} \\
W_2(\mu_K, \pi) &= \tilde{O}\left(\Delta^{e/(2+2e)}\right), \text{ for SGLDLP-F,} \\
W_2(\mu_K, \pi) &= \tilde{O}\left(\Delta^{e/(2+4e)}\right), \text{ for VC SGHMCLP-L,} \\
W_2(\mu_K, \pi) &= \tilde{O}\left(\Delta^{e/(6+6e)}\right), \text{ for VC SGLDLP-L.}
\end{aligned} \tag{5.16}$$

where e is Euler’s number.

Note that $\Delta < 1$ in general, thus Theorem 5.6 suggests that HMC-based low-precision sampling algorithms have better dependence on the quantization error than SGLD-based algorithms for non-log-concave distributions.

Numerical Experiments Similarly to the experiment settings for low-precision SGLD, we consider image tasks CIFAR-10 and CIFAR-100 on the ResNet-18. The simulation results are summarized in Table 5.2. Together with Table 5.1, it shows that across all evaluation metrics, the low-precision SGHMC has either a better or comparable performance with its SGLD counterpart.

5.3 Post-Training Low-Precision Sampling

The previous section focuses on training low-precision Bayesian DNN models directly by SG-MCMC from scratch. In contrast, it is feasible first to train a full-precision (Bayesian) DNN and then obtain a low-precision Bayesian DNN model via additional processing, such as fine-tuning, over the pre-trained model. More relevant content about model compression with quantization can be found in Chapter 41

5.3.1 Quantized Pre-Trained Bayesian models

To convert a pre-trained Bayesian DNN to its low-precision counterpart, one common approach is to quantize the sampled weights yielded by the full-precision posterior sampling (e.g., from SG-MCMC algorithms). In simple words, let $\theta^{(1)}, \dots, \theta^{(T)}, \dots$ be the posterior samples in $\mathbb{R}^P$, then a post-training quantization technique is applied to each sample.

This process seems trivial, but there are multiple factors that need to be considered.

Sampling Thinning Sample thinning refers to the selection of a subset from the original posterior samples. That is, only a proportion of samples will be quantized and used for low-precision Bayesian model average-based inferences, for the sake of computational efficiency. One would also avoid samplers from the burn-in period of posterior sampling to reduce the sampling bias. One common way to include one sample every other f samples (where f denotes the thinning frequency), skipping the first b samples (where b denotes the burn-in period length).

To determine the best thinned sample set, one needs to assess the quality of the samples, say how well the empirical distribution determined by this sample set resembles the true posterior distribution. To this end, we

employ the Kernel Stein Discrepancy [KSD, 652]. KSD is a distance measure between two distributions, where the target distribution is continuously differentiable with an intractable normalization constant. Given a kernel function $k(\cdot, \cdot)$, we define the *Stein Kernel* as

$$k_p(\theta, \vartheta) = \langle \nabla_\theta, \nabla_\vartheta k(\theta, \vartheta) \rangle - \langle \nabla_\theta U(\theta), \nabla_\vartheta k(\theta, \vartheta) \rangle \\ - \langle \nabla_\vartheta U(\vartheta), \nabla_\theta k(\theta, \vartheta) \rangle + \langle \nabla_\theta U(\theta), \nabla_\vartheta U(\vartheta) \rangle k(\theta, \vartheta).$$

Then, the KSD between (the empirical distribution of) a sample $\theta^{(1)}, \dots, \theta^{(T)}$ and the target distribution is

$$\mathbb{S}(\theta^{(1)}, \dots, \theta^{(T)}) = \frac{1}{T^2} \sum_{i,j=1}^T k_p(\theta^{(i)}, \theta^{(j)}).$$

KSD can be used to search for the optimal choices of b and f or the whole subset [1393]. However, it is worth mentioning that the computation of the Stein Kernel scales up quickly with the dimension d , thus it could be very expensive for huge models.

Calibration of Quantization Range and Gap If training low-precision (Bayesian) DNNs from scratch (e.g., Section 5.2), one usually needs to determine the quantization scheme (e.g., the quantization gap Δ in the quantization functions Q^d or Q^s) beforehand. In contrast, when a pre-trained model is available (e.g., a full-precision posterior sample $\theta^{(t)}$), we have direct access to the distribution of its weight values; we can use a representative dataset to approximate the distribution of activations along the forward propagation. This information can help us choose an appropriate quantization scheme to quantize this $\theta^{(t)}$ that fits the computational / memory budgets and has a small impact on its predictive performance. Note that in Section 5.2, we only consider a rounding mechanism that maps a high-precision (e.g., full floating-point value) to its near low-bit fixed-point values. But in general, any mapping from real values to a discrete space can be used. For example, an uniform quantization [827] with W -bit quantization is defined as

$$Q^u(\theta) = \left\lfloor \frac{\text{clip}(\theta, a, b) - a}{s} \right\rfloor s + a, \text{ with } s = (b - a) / (2^W - 1),$$

where $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer. Then a and b , which denote the quantization range, can be set as the minimum and maximum values among entries of the pre-trained model $\theta^{(t)}$.

Parameter Fine-tuning Although a quantized network $Q(\theta^{(t)})$ for some quantization function Q is ready for inference, it may suffer significant performance drops. In such a case, one can first fine-tune the model $\theta^{(t)}$ for a small number of gradient descent steps to compensate for the performance drops caused by quantization, then apply quantization to the fine-tuned model.

Let $f(x; \theta)$ be the neural network model parameterized by the weight vector θ , then the empirical loss is defined as

$$\mathcal{L}(\theta) = - \sum_{(x,y) \in \mathcal{D}} \log p(x, y | \theta) := \sum_{(x,y) \in \mathcal{D}} l(y, f(x; \theta)),$$

for some loss function $l(\cdot, \cdot)$. Instead of running fine-tuning gradient descent on $\mathcal{L}(\theta)$, we design a new neural network model $f^q(\cdot; \theta)$ which approximates $f(\cdot; \theta)$ with quantization applied on weights and activations. For example, Ferianc et al. [515], Jacob et al. [827] insert simulated quantization modules after weights input and every tensor product operation. Therefore, running gradient descent on $\mathcal{L}^q = \sum l(y, f^q(x; \theta))$ initialized from $\theta^{(t)}$, leads to a nearby model whose after-quantization performance is acceptable.

Quantize Pre-trained Variational Bayesian DNNs If a pre-trained variational Bayesian neural network model (refer to Book Part III for details) is given, one may additionally quantize the variational parameters [515, 1571]. Suppose that the variational posterior distribution of one weight entry $\theta_{(i)}$ is $N(\mu, \sigma^2)$, then given a quantization function Q , the quantized posterior sample of $\theta_{(i)}$ is generated by $Q(Q(\mu) + Q(\sigma)\xi)$ where ξ is a standard Gaussian variable. To accommodate performance degradation induced by quantization, one can also incorporate additional fine-tuning for the variational parameters [515], similarly to the procedure discussed in the above paragraph.

5.3.2 Low-Bit Sub-Distribution

As discussed earlier, a pre-trained full-precision model can provide valuable insights into the distribution of weight values and inform the design of a quantization scheme. The SG-MCMC algorithms in Section 5.2 yield posterior samples whose entries belong to an equally spaced discrete space $\mathcal{S} = \{0, \pm\Delta, \pm2\Delta, \dots\}$. When a pre-trained model is available, one can train a flexible discrete space $\mathcal{S}$ from which the posterior samples are drawn. The elements in $\mathcal{S}$ can be full floating-point values; thus, model weights of a neural network achieve low-bit representation by sharing the same full precision values in $\mathcal{S}$. This can be viewed as a generalization of sparse DNN, which encourages network weights to share a common value of 0.

Given a pre-trained DNN model θ , a straightforward method is to implement a clustering algorithm over the entries of θ and obtain the cluster centers $\mathcal{S} = \{q_1, \dots, q_M\}$. Then we would like to restrict the posterior samples to $\mathcal{S}$. To accomplish this, we can adapt the mean-field variational Bayes (refer to Chapter 12 for details) and define the discrete variational distribution as

$$q(\theta) = \prod_{i=1}^P \sum_{m=1}^M p_{im} \mathbf{1}(\theta_{(i)} = q_m), \quad \theta = (\theta_{(1)}, \dots, \theta_{(P)}) \quad (5.17)$$

where variational parameters satisfy $\sum_m p_{im} = 1$. To facilitate the training of variational inference, we can introduce parametric modelling for p_{im} (e.g., p_{im} relates to $|\theta_{(i)} - q_m|$, where $\theta_{(i)}$ is the pre-trained weight). To enhance the model performance, we can also allow $q_1, \dots, q_M$ to update in the variational training procedure [see e.g., 439].

It is worth mentioning that no actual low-precision sampling happens in the variational training process since variational inference boils down to an optimization problem for the variational parameters. Once these parameters are learned, the (approximate) posterior samples are easily drawn from (5.17).

5.4 Conclusions and Discussions

This chapter introduced low-precision sampling algorithms that are useful for modern Bayesian deep learning applications, including SG-MCMC-based direct low-precision samplers and post-training low-precision sampling techniques. These low-precision methods are crucial for deploying large models on resource-constrained hardware and improving Bayesian inference speed.

As discussed in the introduction section, one main reason for pursuing low-precision arithmetics is to reduce the memory and computation costs for DNN training or DNN inference. Beyond this, the low-precision representation is also helpful to reduce communication cost for scenarios where frequent information transfer among several parties is required, but only with limited bandwidth. One such emerging research field is federated learning (FL) [859]. Federated learning is a learning paradigm where multiple parties (e.g., hospitals or edge devices) collaboratively train a global model while keeping the local training data private and decentralized. In a federated SGLD algorithm, each party computes the gradient based on its local data, and all local gradients are shared and aggregated via communication. The aggregated gradient is then used in the SGLD update (5.1) of the global model. Low-precision representations can be used to compress the gradient communications. Vono et al. [1688] utilized a stochastic rounding mechanism and studied how quantization error impacts the convergence of the global Bayesian model.

The theorems and algorithms in this chapter are mostly depicted regarding the theme that quantization of the weight parameter θ suffices. However, neural networks have a multilayer structure, and quantization can be employed on each layer along the forward propagation process (i.e., online conversion of activation values to low-precision representations). In addition, normalization layers, such as Batch Normalization and Layer Normalization, are often sensitive to quantization error and may require different techniques (e.g., BatchNorm folding) to mitigate the impact of quantization. These issues are less explored and discussed in the field of Bayesian low-precision DNNs. Interested readers may refer to Gholami et al. [615], Kim et al. [889].

Chapter 6

Discrete sampling

Many tasks within deep learning involve sampling over discrete spaces. Such sampling tasks are distinct from sampling over continuous spaces: while the primary frameworks of sampling—Markov chain Monte Carlo (MCMC), rejection sampling, and importance sampling—introduced in Chapter 1 can be applied to discrete spaces by adjusting the proposal distribution, this adjustment is often highly non-trivial. In this chapter, we focus on high-dimensional targets whose unnormalised log density can be evaluated but whose state spaces are too large to enumerate. Such targets arise in binary energy-based models, categorical latent-variable models, and neural models over combinatorial structures.

6.1 Introduction

Discrete sampling is notably difficult due to the sparse, discontinuous, and combinatorial nature of discrete spaces. If a state consists of d categorical variables with K possible values each, its state space may contain K^d configurations. In principle, one could evaluate the unnormalised density at every configuration, normalise the resulting weights, and sample from the corresponding categorical distribution. This enumeration-based approach becomes infeasible as the dimension grows. Standard local MCMC methods avoid full enumeration, but target-independent proposals can mix slowly because they do not distinguish promising moves from moves into low-density regions.

As a running example, consider a binary energy-based model with $x \in \{0, 1\}^d$ and unnormalised log density $U(x)$. The model assigns a score to each of its 2^d possible configurations, making enumeration infeasible for large d . Nevertheless, $U(x)$, and often the gradient of a differentiable extension of U , can be evaluated at any given configuration. We use this binary model to illustrate the methods below, although the framework also applies to the broader categorical and combinatorial settings described above.

Gradients are a staple of sampling within deep learning, as illustrated by the continuous-space Langevin methods in Section 1.5.1. Although a function defined only on a discrete space has no intrinsic gradient, many deep-learning targets are represented through a differentiable function on a continuous ambient space and evaluated at binary, categorical, or other discrete states. The gradient of this extension can provide a cheap source of local information about promising discrete moves.

We begin with an uninformed local MCMC proposal and show how pointwise-informed and locally balanced proposals incorporate target-specific information. We then use this framework to develop gradient-based discrete samplers, first for single-coordinate moves and then for parallel multi-coordinate moves. Finally, we discuss methods designed for larger-scale exploration.

6.2 Notation

We assume a discrete state space $\mathcal{X}$. We denote a state by $x = (x_1, \dots, x_d) \in \mathcal{X}$, where each coordinate can take one of K possible values. Throughout the chapter, π denotes the normalised target distribution and U its unnormalised log density. Thus our target is

$$\pi(x) = \frac{1}{Z} \exp(U(x)), \quad Z = \sum_{x' \in \mathcal{X}} \exp U(x') \quad (6.1)$$

The normalising constant or partition function Z in Equation (6.1) generally requires a sum over as many as K^d states. This makes enumeration-based direct sampling computationally infeasible in high dimensions. MCMC

can nevertheless operate on differences such as $U(x') - U(x)$, in which Z cancels.

6.3 Locally Balanced Proposals

A simple Metropolis–Hastings approach proposes a nearby state using a target-independent kernel $K_\sigma(x' | x)$. For example, the kernel might choose a coordinate uniformly and replace its value. Such a proposal is inexpensive, but it assigns the same proposal probability to moves towards high- and low-density states. Large moves may explore quickly but often reach low-density states and are rejected; small moves are more likely to be accepted but can produce highly correlated samples and traverse the state space slowly.

For the running binary example, the simplest local kernel selects one of the d bits uniformly and flips it, without using the values of U to prioritise the available moves.

Zanella [1831] introduces a framework that addresses this trade-off by weighting the candidate moves from K_σ using target-density information. The resulting pointwise-informed proposal is

$$P_{g,\sigma}(x' | x) = \frac{g\left(\frac{\pi(x')}{\pi(x)}\right) K_\sigma(x' | x)}{Z_{g,\sigma}(x)}. \quad (6.2)$$

The pointwise-informed proposal in Equation (6.2) consists of the target-density ratio, a symmetric base kernel or weight K_σ whose scale controls the size of candidate moves, a balancing function g that weights moves using the density ratio, and the state-dependent normaliser $Z_{g,\sigma}(x)$. The base kernel may be specified only up to a state-independent multiplicative constant, because that constant cancels through $Z_{g,\sigma}(x)$. We assume $K_\sigma(x' | x) = K_\sigma(x | x')$. The limit $\sigma \rightarrow 0$ represents a family of kernels that becomes increasingly concentrated at the current state.

The roles of invariance and ergodicity in MCMC are reviewed in Section 1.5, while detailed balance and the Metropolis–Hastings construction are discussed in Section 1.5.1. In the present setting, $P_{g,\sigma}$ is reversible with respect to $\pi_{g,\sigma}$ if it satisfies detailed balance,

$$\pi_{g,\sigma}(x) P_{g,\sigma}(x' | x) = \pi_{g,\sigma}(x') P_{g,\sigma}(x | x').$$

Detailed balance is sufficient for $\pi_{g,\sigma}$ to be invariant under the transition. Invariance alone does not ensure convergence from an arbitrary initial state. For a discrete chain, irreducibility means that every state with positive target probability can eventually be reached from every admissible initial state, while aperiodicity means that the chain is not constrained to return according to a fixed cycle. Together with invariance and the relevant regularity conditions, these properties yield an ergodic chain. The general role of ergodicity in MCMC is reviewed in Section 1.5.

A pointwise-informed proposal is called *locally balanced* if the transition kernel $P_{g,\sigma}$ is reversible with respect to some distribution $\pi_{g,\sigma}$ such that $\pi_{g,\sigma}$ converges weakly to π as $\sigma \rightarrow 0$ [1831]. In plain terms, as the proposal becomes increasingly local, the invariant distribution of the uncorrected informed proposal approaches the desired target. Zanella [1831] demonstrates that a pointwise-informed proposal is locally balanced if and only if the balancing function satisfies $g(t) = tg(1/t)$.

If this property holds for the function g , then the locally balanced proposal will be reversible with respect to the following distribution without any correction:

$$\pi_{g,\sigma}(x) = \frac{\pi(x) Z_{g,\sigma}(x)}{\sum_{x' \in \mathcal{X}} \pi(x') Z_{g,\sigma}(x')}.$$

Here, $Z_{g,\sigma}$ is defined as follows:

$$Z_{g,\sigma}(x) = \sum_{x' \in \mathcal{X}} g\left(\frac{\pi(x')}{\pi(x)}\right) K_\sigma(x' | x).$$

For a local kernel satisfying $K_\sigma(x' | x) \rightarrow \mathbf{1}\{x' = x\}$ and the conventional normalisation $g(1) = 1$, we have $Z_{g,\sigma}(x) \rightarrow 1$. Consequently, $\pi_{g,\sigma}$ converges to π as $\sigma \rightarrow 0$ [1831].

At a non-zero scale, $\pi_{g,\sigma}$ need not equal π , so a Metropolis–Hastings correction is still required for exact sampling. Local balance makes the uncorrected proposal close to the target in the small-scale regime, which can improve the acceptance rate, while a non-zero σ is required to produce non-degenerate movement across the state space.

Zanella [1831] shows that locally balanced proposals can improve efficiency over uninformed local proposals in the asymptotic regimes considered there. Constructing an exact informed proposal nevertheless requires evaluating

the density ratio for every candidate. Full enumeration can involve K^d states; even a Hamming-one neighbourhood contains $d(K - 1)$ candidates. The gradient-based methods below replace these repeated neighbouring-state evaluations with one gradient evaluation at the current state.

6.4 Gradient-Based Discrete Sampling

A function specified only on a discrete state space does not have an intrinsic gradient. Gradient-based discrete samplers therefore assume that the unnormalised log density U is supplied with, or represented through, a differentiable extension to a continuous ambient space. For example, a neural network may accept a continuous vector representation even though the sampler evaluates it only at binary or one-hot encoded states. This extension need not be unique, and its gradient is useful only insofar as it predicts changes between the discrete states of interest.

Given such an extension, the log-density difference for a nearby state can be approximated by

$$U(x') - U(x) \approx \nabla U(x)^\top (x' - x).$$

This approximation is most reliable when the extension is sufficiently smooth and x' is close to x ; differentiability alone does not guarantee accuracy if the function changes sharply between neighbouring states. Since $\log \pi(x) = U(x) - \log Z$, their gradients are identical and the unknown normalising constant is not required. One forward and backward pass at x then supplies approximate scores for all local candidate moves, at the expense of Taylor-approximation error.

6.4.1 Gibbs with Gradient

Grathwohl et al. [659] first introduces this notion of gradient-based discrete sampling through the Gibbs-With-Gradient algorithm. This algorithm can be seen as a version of the standard Gibbs algorithm where the updated coordinate is selected based on the gradient.

Returning to the general locally balanced sampler, we can derive the Gibbs-With-Gradient update by choosing an uninformed kernel that uniformly selects a state differing from the current state in exactly one coordinate. Define the Hamming-one neighbourhood by

$$H(x) = \left\{ x' \in \mathcal{X} : \sum_{i=1}^d \mathbf{1}\{x'_i \neq x_i\} = 1 \right\}.$$

In the running binary example, $H(x)$ contains exactly the d states obtained by flipping one bit of x .

For this fixed-scale update, the uniform kernel is

$$K_\sigma(x' | x) = \frac{\mathbf{1}\{x' \in H(x)\}}{|H(x)|}.$$

When every coordinate has K possible categories, $|H(x)| = d(K - 1)$. This factor is constant across candidate moves and is therefore absorbed into the proportionality below.

Next, we choose the balancing function $g(t) = \sqrt{t}$. Before approximation, this yields the exact informed proposal

$$q_{\text{exact}}(x' | x) \propto \mathbf{1}\{x' \in H(x)\} \sqrt{\frac{\pi(x')}{\pi(x)}}$$

Using the unnormalised log density U from Equation (6.1), this proposal can be expressed without evaluating Z :

$$q_{\text{exact}}(x' | x) \propto \mathbf{1}\{x' \in H(x)\} \exp\left(\frac{1}{2}(U(x') - U(x))\right).$$

Replacing the exact log-density difference by its first-order Taylor approximation defines the Gibbs-With-Gradient proposal:

$$q_{\text{GWG}}(x' | x) \propto \mathbf{1}\{x' \in H(x)\} \exp\left(\frac{1}{2}\nabla U(x)^\top (x' - x)\right).$$

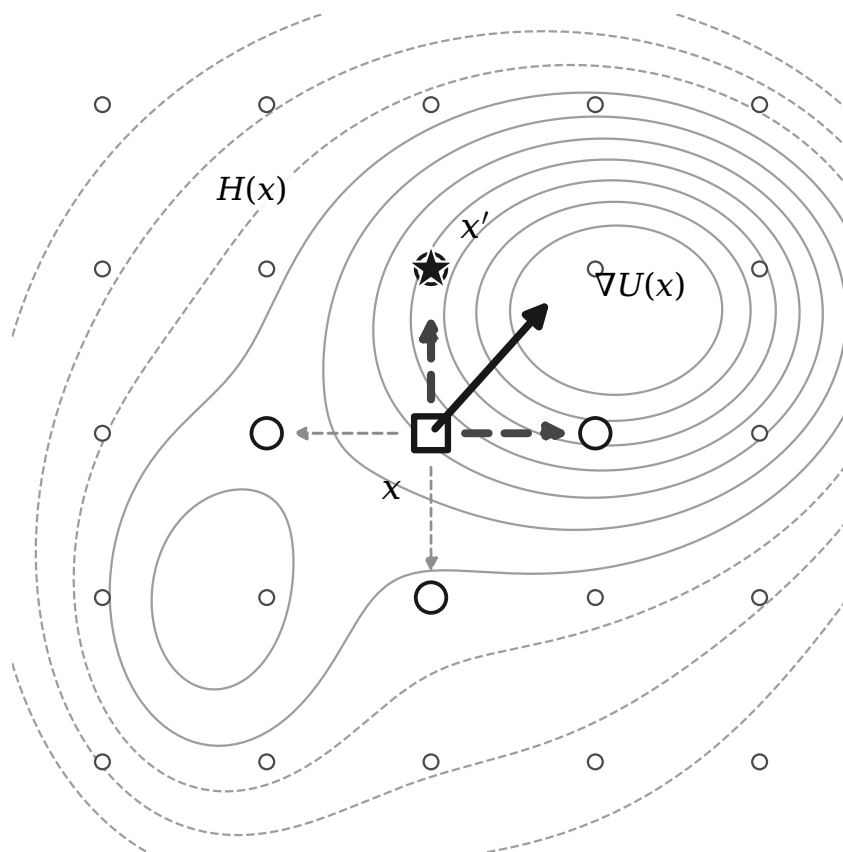


Figure 6.1: Gradient-informed sampling on a discrete state space. The gradient $\nabla U(x)$ of a differentiable extension provides local geometric information at the current state x , which induces different proposal weights over its Hamming-one neighbourhood $H(x)$. The starred state x' represents a draw from this proposal. Adapted from Grathwohl et al. [659].

Figure 6.1 illustrates this Gibbs-With-Gradient construction. The gradient at x assigns different approximate weights to the states in $H(x)$, and x' is then sampled from the resulting proposal over those discrete neighbours.

This can be computed quickly by viewing the update as the selection of a coordinate to modify. In the binary case, this update is always a bit flip. Here, Softmax normalises a vector of real-valued scores into probabilities, and Cat denotes the categorical distribution with those probabilities. The resulting proposal is

$$q_{\text{GWG}}(\cdot | x) = \text{Cat} \left(\text{Softmax} \left(-\frac{1}{2}(2x - 1) \odot \nabla U(x) \right) \right).$$

The categorical case requires selecting both a coordinate and its proposed category. Let c_i denote the current category of coordinate i . We represent each categorical coordinate by a one-hot vector when evaluating the gradient. For every alternative category $j \neq c_i$, define the approximate log-probability difference

$$\tilde{d}(x)_{i,j} = \nabla_{x_{i,j}} U(x) - \nabla_{x_{i,c_i}} U(x).$$

The first term measures the local effect of assigning category j to coordinate i , while the second corresponds to its current category.

There are $d(K-1)$ possible coordinate–category updates. We define a categorical proposal over these updates by

$$q_{\text{GWG}}(i, j | x) = \text{Softmax}_{i \in [d], j \neq c_i} \left(\frac{1}{2} \tilde{d}(x)_{i,j} \right).$$

After sampling $(i, j) \sim q_{\text{GWG}}(\cdot, \cdot | x)$, the proposed state x' is obtained by changing coordinate i from c_i to j and leaving all other coordinates unchanged. The full categorical procedure is given in Algorithm 14.

Algorithm 14: Categorical Gibbs With Gradient Sampler

Inputs : initial categorical state $x = (x_1, \dots, x_d) \in \mathcal{X}$; unnormalised log-density U

Outputs: sample sequence $\mathcal{S}$

$\mathcal{S} \leftarrow ()$

for each iteration **do**

 compute $\tilde{d}(x)$ and $q_{\text{GWG}}(\cdot, \cdot | x)$

 sample $(i, j) \sim q_{\text{GWG}}(\cdot, \cdot | x)$

$c_i \leftarrow$ current category of coordinate i

$x' \leftarrow \text{replace}(x, i, j)$

 compute the reverse-move probability $q_{\text{GWG}}(i, c_i | x')$

 accept x' with probability

$$\min \left\{ 1, \exp(U(x') - U(x)) \frac{q_{\text{GWG}}(i, c_i | x')}{q_{\text{GWG}}(i, j | x)} \right\}$$

 update $x \leftarrow x'$ if accepted

 append x to $\mathcal{S}$

return $\mathcal{S}$

Because the gradient-based proposal uses a first-order approximation and is generally asymmetric, it does not by itself preserve the target distribution. We therefore apply the standard Metropolis–Hastings acceptance correction, described in Section 1.5.1, which ensures that the target distribution is invariant. Under the ergodicity conditions stated above, the resulting Markov chain converges to the target distribution.

6.4.2 Discrete Langevin Proposal

Gibbs-With-Gradient can outperform uninformed single-coordinate proposals by using the gradient to prioritise promising updates. Its Hamming-one restriction also keeps the Taylor approximation local, but only one coordinate can change per iteration, which may produce strongly correlated samples in high dimensions. Zhang et al. [1855] addresses this limitation with the Discrete Langevin Proposal (DLP), which retains gradient-based directionality while adding a distance penalty that permits multiple coordinates to be proposed simultaneously.

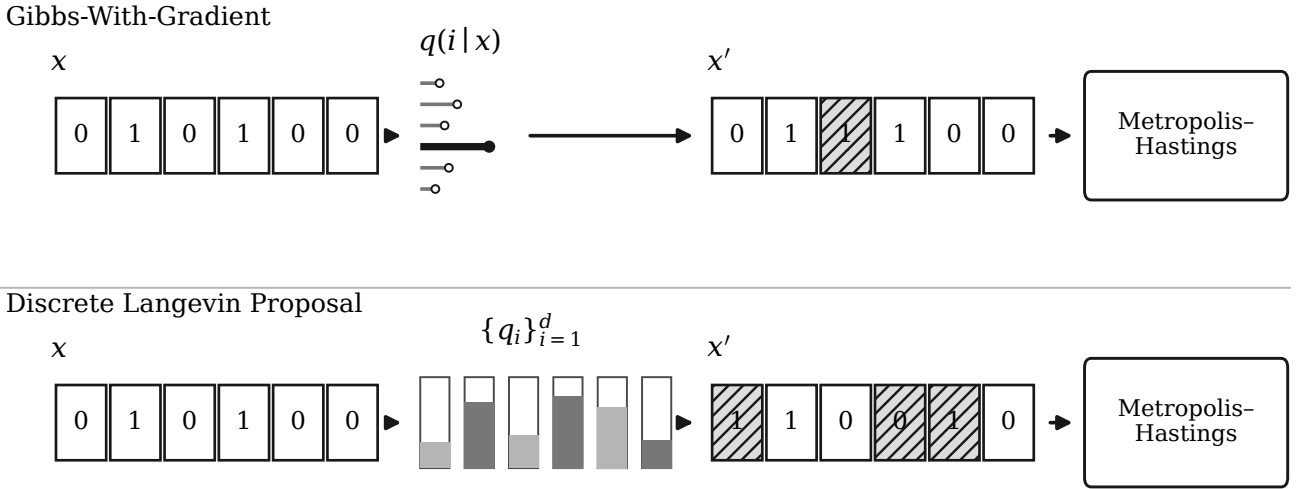


Figure 6.2: Comparison of gradient-based discrete proposals. Gibbs-With-Gradient samples one coordinate from $q_{\text{GWG}}(i | x)$ and proposes a state that differs in that coordinate. The Discrete Langevin Proposal constructs coordinate-wise distributions $\{q_i\}_{i=1}^d$ and can propose several changes in parallel. The displayed Metropolis–Hastings correction gives the adjusted variants; omitting it yields DULA in the discrete Langevin case. Based on Grathwohl et al. [659] and Zhang et al. [1855].

In the running binary example, Gibbs-With-Gradient proposes at most one bit flip, whereas DLP constructs a proposal for every bit and may change several bits in the same iteration.

To derive the proposal, consider the discrete analogue of a Gaussian random-walk kernel,

$$K_{\sigma}(x' | x) \propto \exp\left(-\frac{1}{2\sigma}\|x' - x\|_2^2\right).$$

Combining this kernel with a first-order approximation of the locally balanced weight gives the DLP

$$q(x' | x) \propto \exp\left(\frac{1}{2}\nabla U(x)^{\top}(x' - x) - \frac{1}{2\sigma}\|x' - x\|_2^2\right). \quad (6.3)$$

The linear term in Equation (6.3) favours moves predicted to increase U , while the quadratic term penalises moves far from the Taylor expansion point. The step size σ controls the strength of this penalty.

Because the state space factorises as $\mathcal{X} = \prod_{i=1}^d \mathcal{X}_i$ and the exponent in Equation (6.3) is a sum of coordinate-wise terms, the proposal factorises as $q(x' | x) = \prod_{i=1}^d q_i(x'_i | x_i)$. Each factor is the categorical distribution

$$q_i(x'_i | x_i) = \text{Cat}\left(\text{Softmax}_{x'_i \in \mathcal{X}_i} \left[\frac{1}{2}\nabla_i U(x)(x'_i - x_i) - \frac{(x'_i - x_i)^2}{2\sigma} \right]\right).$$

Thus, larger values of σ permit more distant proposals, whereas smaller values concentrate the proposal near the current state.

While this may look similar to the proposal formula for Gibbs with Gradient, it is important to note that in DLP, the softmax is computed per coordinate over the potential classes that can be realised. In Gibbs with Gradient, the softmax is over the entire sample to pick only one coordinate. This allows for the parallel update of all coordinates.

The factorisation makes it possible to propose all coordinates in parallel after one gradient evaluation. Although the proposal factors coordinate-wise conditional on the current state, the gradient of the joint log density still guides each coordinate using information from the full current configuration. A Metropolis–Hastings correction can then account for the remaining approximation error and dependence structure. Figure 6.2 contrasts this parallel update with the single-coordinate update used by Gibbs-With-Gradient.

Using DLP yields two samplers: the discrete Metropolis-adjusted Langevin algorithm (DMALA), which includes a Metropolis–Hastings correction, and the discrete unadjusted Langevin algorithm (DULA), which omits it. DULA generally has asymptotic bias. For log-quadratic targets, however, Zhang et al. [1855] show that this

bias vanishes as $\sigma \rightarrow 0$; they also bound the bias for targets whose gradients are close to those of a log-quadratic distribution.

In terms of implementation, the cost difference between DULA and DMALA is negligible: while DMALA incorporates a Metropolis–Hastings correction that requires the computation of both $q(\cdot | x)$ and $q(\cdot | x')$, caching means that only one new proposal distribution is computed per step after initialisation. The cached version of DMALA is given in Algorithm 15.

Algorithm 15: DMALA with Cache

Inputs : initial state $x = (x_1, \dots, x_d) \in \mathcal{X}$; unnormalised log-density U

Outputs: sample sequence $\mathcal{S}$

$\mathcal{S} \leftarrow ()$

compute $q(\cdot | x)$ and cache proposal

for each iteration do

 sample $x' \sim q(\cdot | x)$

 compute $q(\cdot | x')$ using DLP proposal

 accept x' with probability

$$\min \left\{ 1, \exp(U(x') - U(x)) \frac{q(x | x')}{q(x' | x)} \right\}$$

if accepted then

 update cached proposal to $q(\cdot | x')$

$x \leftarrow x'$

 append x to $\mathcal{S}$

return $\mathcal{S}$

The core logic lies in the fact that the x' of the current iteration is either rejected or becomes the x for the next sampling step. In the case of rejection, the proposal function $q(\cdot | x)$ does not change as x remains the same. In the case of acceptance, the proposal step simply gets updated to match the updated x' by using $q(\cdot | x') \rightarrow q(\cdot | x)$, since this proposal must be computed anyway for the accept/reject step. Thus at each step, only one proposal function must be computed.

6.5 Beyond Locally Balanced Proposals

Locally balanced proposals offer a convenient framework for incorporating gradients into discrete samplers, but their defining approximation concerns the small-scale limit $\sigma \rightarrow 0$. At finite scales, local proposals may take a long time to move between separated modes, necessitating longer chains or multiple chains.

For example, if the binary target places high probability on two configurations separated by many low-probability intermediate states, a sequence of single-bit proposals may struggle to move from one mode to the other.

This motivates proposals designed for non-local movement. Zanella [1831] already considers $g(t) = t$, calling the resulting proposal the *naively informed* choice because it has the form $P_{g,\sigma}(x' | x) \propto K_\sigma(x' | x)\pi(x')$. Building on this observation, Sun et al. [1574] formalise the large-scale interpretation: they call a proposal family *globally balanced* if the invariant distribution of its uncorrected informed proposal converges to π as $\sigma \rightarrow \infty$. They assume that the base kernel becomes uniform in this limit. Since a state-independent constant cancels in the normalisation of the informed proposal, they write this limit as $K_\sigma(x' | x) \rightarrow 1$; for a normalised kernel on a finite state space, the corresponding limit is $1/|\mathcal{X}|$. For $g(t) = t$, the limiting proposal is then

$$\lim_{\sigma \rightarrow \infty} P_{g,\sigma}(x' | x) = \frac{\pi(x')/\pi(x)}{\sum_{z \in \mathcal{X}} \pi(z)/\pi(x)} = \pi(x').$$

Thus the proposal becomes an independent draw from the target itself and is globally balanced in the sense of Sun et al. [1574]. Notice that $g(t) = t$ does not satisfy the locally balanced identity $g(t) = tg(1/t)$; local and global balance refer to different limiting regimes.

6.5.1 Any-scale Balanced Sampler

The any-scale balanced sampler introduced by Sun et al. [1574] makes two distinct modifications for proposals outside the local regime. First, it adjusts the density-ratio weighting to match the proposal scale. The family $g(t) = t^\beta$, with $\beta \in [1/2, 1]$, interpolates between the locally balanced choice $g(t) = \sqrt{t}$ as $\sigma \rightarrow 0$ and the globally balanced choice $g(t) = t$ as $\sigma \rightarrow \infty$. This scale-dependent relationship is illustrated schematically in Figure 6.3(a).

Adjusting β changes how strongly the proposal favours high-density candidates, but it does not reduce the error in a first-order approximation of $U(x') - U(x)$. The method therefore makes a second modification: it introduces a quadratic approximation intended to remain informative over larger moves. The log-density difference is approximated by

$$U(x') - U(x) \approx (x' - x)^\top \nabla U(x) + \frac{1}{2}(x' - x)^\top W(x' - x), \quad (6.4)$$

where W is a symmetric matrix shared across states. Setting $W = \nabla^2 U(x)$ would recover the usual second-order Taylor approximation locally, but repeatedly computing a Hessian is expensive. Instead, the sampler estimates a global W from pairs (x_n, x'_n) collected during burn-in:

$$\widehat{W} = \arg \min_{W=W^\top} \sum_n \|W(x'_n - x_n) - (\nabla U(x'_n) - \nabla U(x_n))\|_2^2.$$

Thus W is fitted to explain observed changes in the gradient and acts as a global curvature approximation. Scale-dependent balancing and curvature approximation address different problems: β controls the density-ratio weighting, whereas W improves the approximation used to score non-local candidates.

Substituting the quadratic approximation in Equation (6.4) and $g(t) = t^\beta$ into the informed proposal gives

$$q(x' | x) \propto \exp \left(\beta \left[(x' - x)^\top \nabla U(x) + \frac{1}{2}(x' - x)^\top W(x' - x) \right] - \frac{1}{2\sigma} \|x' - x\|_2^2 \right). \quad (6.5)$$

The quadratic term captures cross-coordinate interactions, but it also prevents direct coordinate-wise sampling. Sun et al. [1574] therefore introduce an augmented proposal with a Gaussian auxiliary variable. The augmented construction is used directly with the Metropolis–Hastings correction below.

Choose a diagonal positive semi-definite matrix D such that $W + D$ is positive semi-definite, and draw

$$Y | x \sim \mathcal{N} \left((W + D)^{1/2} x, I \right).$$

The matrix square root can be computed by an eigendecomposition or Cholesky factorisation. Conditional on Y , define

$$q(x' | x, Y) \propto \exp \left(\beta x'^\top \left[\nabla U(x) - Wx + (W + D)^{1/2} Y \right] - \frac{1}{2} x'^\top \left(\beta D + \frac{1}{\sigma} I \right) x' \right),$$

and sample x' from this distribution. Because D is diagonal and the state space factorises, this conditional proposal factorises across coordinates. Equations (6.5) and the augmented proposal express, respectively, the motivating quadratic approximation and the auxiliary-variable transition implemented by the sampler. The Metropolis–Hastings step accounts for the resulting augmented proposal exactly.

The Metropolis–Hastings acceptance probability for the augmented proposal is

$$A(x, x', Y) = \min \left\{ 1, \exp(U(x') - U(x)) \frac{q(Y | x') q(x | x', Y)}{q(Y | x) q(x' | x, Y)} \right\}.$$

The diagonal matrix D controls the conditional proposal variance. Sun et al. [1574] consider two semidefinite programs for choosing it. The AB-max variant minimises the largest eigenvalue of $W + D$:

$$(D^*, \lambda^*) = \arg \min_{D, \lambda} \lambda \quad \text{subject to} \quad D \succeq 0, \quad 0 \preceq W + D \preceq \lambda I.$$

This criterion focuses on the worst spectral direction. The AB-trace variant instead solves

$$D^* = \arg \min_D \text{tr}(D) \quad \text{subject to} \quad D \succeq 0, \quad W + D \succeq 0.$$

The motivation for this objective comes from the continuous quadratic surrogate studied by Sun et al. [1574] in the special case $\sigma = \infty$ and $\beta = 1$. In that surrogate, the covariance of the proposal increment is $2D^{-1}$, so minimising

the trace of D encourages dispersion across coordinates. This covariance identity does not hold literally for the general discrete categorical proposal, whose dispersion also depends on β , σ , and the state space. The authors report that AB-trace performs better empirically and use it as their preferred variant.

The proposal parameters β and σ are correlated and target-dependent. Sun et al. [1574] therefore adapt them by alternating coordinate-wise updates that seek to maximise the empirical average jump distance. For one parameter type $a \in \{\sigma, \beta\}$, let ϕ denote its current value and define the feasible sets $\mathcal{I}_\sigma = (0, \infty)$ and $\mathcal{I}_\beta = [1/2, 1]$. With the other parameter held fixed, the adaptation block compares ϕ with feasible multiplicative perturbations of it over three consecutive chain segments, as shown in Algorithm 16.

Algorithm 16: Any-scale Adaptation Block

Inputs : parameter type $a \in \{\sigma, \beta\}$; current value $\phi \in \mathcal{I}_a$; adaptation rate $\gamma \in (0, 1)$; initial state $x^{(0)}$
Outputs: updated parameter ϕ' ; samples $x^{(1)}, \dots, x^{(3N)}$

```

 $\phi_0 \leftarrow \phi$ 
 $\phi_+ \leftarrow \text{proj}_{\mathcal{I}_a}(\phi(1 + \gamma))$ 
 $\phi_- \leftarrow \text{proj}_{\mathcal{I}_a}(\phi(1 - \gamma))$ 
use  $\phi_0$  to compute the proposal and collect  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ 
compute  $d_0 \leftarrow \sum_{i=1}^N \|x^{(i)} - x^{(i-1)}\|_1$ 
use  $\phi_+$  to sample  $x^{(N+1)}, \dots, x^{(2N)}$  using the proposal
compute  $d_+ \leftarrow \sum_{i=1}^N \|x^{(N+i)} - x^{(N+i-1)}\|_1$ 
use  $\phi_-$  to sample  $x^{(2N+1)}, \dots, x^{(3N)}$  using the proposal
compute  $d_- \leftarrow \sum_{i=1}^N \|x^{(2N+i)} - x^{(2N+i-1)}\|_1$ 
if  $\max(d_0, d_+, d_-) = d_+$  then
|  $\phi' \leftarrow \phi_+$ 
else if  $\max(d_0, d_+, d_-) = d_-$  then
|  $\phi' \leftarrow \phi_-$ 
else
|  $\phi' \leftarrow \phi_0$ 
return  $\phi'$  and  $x^{(1)}, \dots, x^{(3N)}$ 

```

The three segments in Algorithm 16 are consecutive: the second begins from $x^{(N)}$ and the third from $x^{(2N)}$. The full adaptation alternates a block for $\phi = \sigma$ and a block for $\phi = \beta$. If neither parameter changes during an outer iteration, the update rate is reduced geometrically, $\gamma \leftarrow \rho\gamma$ for a decay factor $\rho \in (0, 1)$; otherwise the selected parameter values are retained for the next iteration. Because adaptation based on short-run jump distances can be noisy during early mixing, this procedure is intended for burn-in rather than continued adaptation after samples are retained for inference.

Through the combination of a quadratic approximation of the log-probability difference $U(x') - U(x)$ and the adaptive scheme for β and σ , the any-scale sampler interpolates between locally and globally balanced proposals.

6.5.2 Automatic Cyclical Sampling

One common issue with gradient-based samplers for continuous spaces is exploration: as the gradient points in the direction of the closest mode, it gives no information regarding further, more distant modes. This can hinder the accuracy of the sampler in terms of approximating the target distribution.

The same problem applies to gradient-based discrete samplers. Unlike a continuous state space, a discrete space has no arbitrarily small non-trivial perturbations: even the most local move must change at least one coordinate. This granularity can make the trade-off between conservative, high-acceptance moves and larger exploratory moves especially pronounced. Interpolating between locally and globally balanced proposals is therefore a natural way to vary the proposal scale when large movements may be required to escape a local mode.

In the continuous case, many approaches have been proposed for handling the mode-escape problem. One popular approach is cyclical SG-MCMC, discussed in Chapter 2. A cyclical schedule gradually reduces the step-size of a gradient-based sampler from very large magnitudes to small magnitudes and repeats for several cycles. The large steps at the beginning allow for more exploratory movements, which enable mode discovery; and the smaller step-sizes at the end of the cycle allow for exploitation of the mode. The repetition of the cycles allows for the discovery and exploitation of many modes, which is required for effective characterisation of multi-modal distributions.

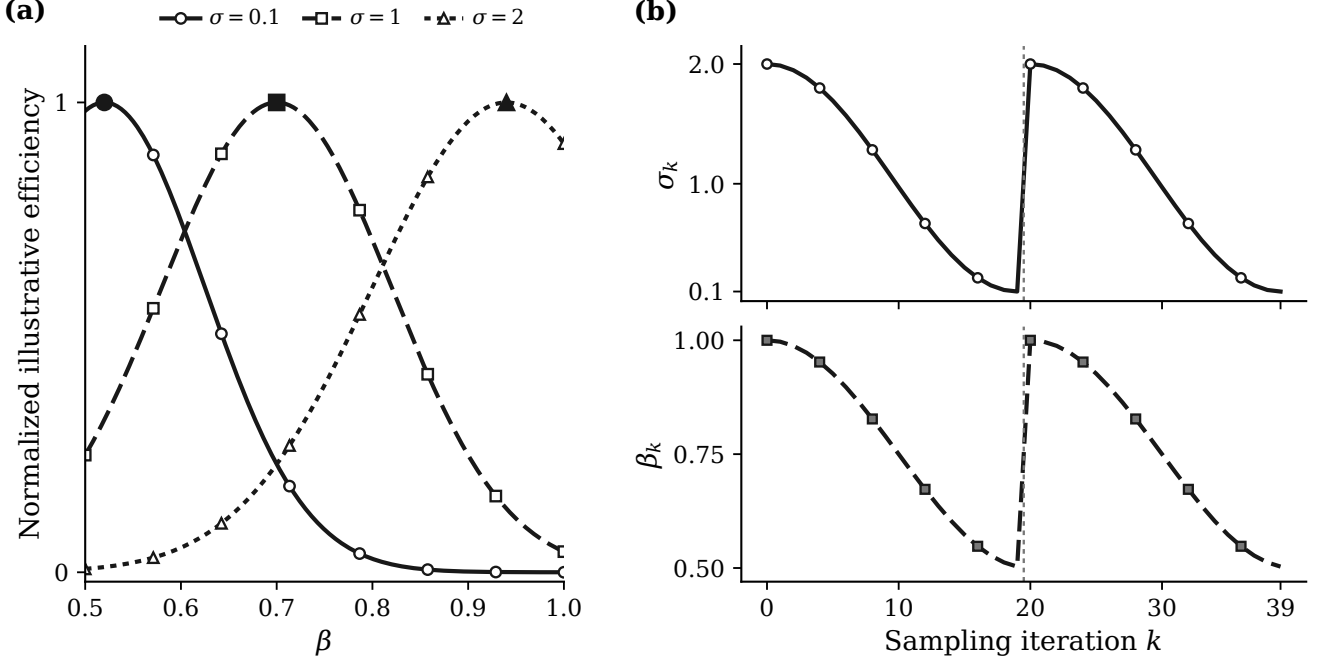


Figure 6.3: Proposal scale, balancing, and cyclical scheduling. Panel (a) schematically illustrates that the preferred balancing exponent increases with the proposal scale: small-scale proposals favour β near $1/2$, whereas larger-scale proposals favour values closer to 1. The curves show normalised illustrative efficiency rather than empirical measurements. Panel (b) shows coupled schedules for σ_k and β_k over two cycles, progressing from large proposals with more globally balanced weighting to small, locally balanced proposals within each cycle. Panel (a) is adapted conceptually from Sun et al. [1574]; panel (b) is adapted from Pynadath et al. [1350].

Building on this intuition, Pynadath et al. [1350] introduce a sampler that cycles jointly through step sizes and balancing exponents. Varying σ alone can produce poorly balanced proposals and low acceptance probabilities, so the method uses the generalised DLP

$$q_i(x'_i | x) = \text{Cat} \left(\text{Softmax}_{x'_i \in \mathcal{X}_i} \left[\beta \nabla_i U(x)(x'_i - x_i) - \frac{(x'_i - x_i)^2}{2\sigma} \right] \right). \quad (6.6)$$

Setting $\beta = 1/2$ in Equation (6.6) recovers the standard DLP. Within each cycle, large values of σ are paired with values of β closer to 1 to encourage global exploration; as σ decreases, β moves towards $1/2$ and the proposal becomes more local.

The automatic tuning procedure begins with user-specified endpoint exponents $\beta_{\min}$ and $\beta_{\max}$ and a target acceptance rate. Short burn-in runs determine corresponding endpoint step sizes $\sigma_{\min}$ and $\sigma_{\max}$. These endpoints define the cyclical step-size schedule, following the continuous-space construction of Zhang et al. [1854].

For each step size σ_k , the corresponding β_k is selected to maximise an empirical acceptance-rate objective subject to $\beta_k \in [1/2, \beta_{k-1}]$. This monotonic constraint reflects the transition from large moves with more globally balanced weighting at the beginning of a cycle to small, locally balanced moves at its end, while reducing the cost of searching for the full schedule. Figure 6.3(b) shows the coupled schedules used to move repeatedly from global exploration to local characterisation.

6.6 Conclusion

Gradient information can make sampling in high-dimensional discrete spaces substantially more efficient, even though the state space itself is not differentiable. Locally balanced proposals provide the organising principle: they combine an uninformed local kernel with target-specific information and use a Metropolis–Hastings correction to retain the desired target distribution. Gibbs-With-Gradient uses a first-order approximation to choose an informed single-coordinate update, whereas DLP factorises a Langevin-like proposal so that many coordinates can be proposed in parallel.

The appropriate proposal depends on the scale of movement required. Small, locally balanced moves can characterise a mode accurately but may mix slowly between separated modes. Any-scale balancing adapts the balancing exponent and adds a global curvature approximation for more effective non-local moves. Automatic cyclical sampling instead repeats a schedule from large exploratory proposals to small exploitative proposals, pairing each step size with a suitable balancing exponent.

In the running binary example, this progression moves from informed single-bit proposals, through parallel multi-bit updates, to methods that adapt the scale and weighting of those updates to improve movement between separated modes.

These methods trade additional structure and tuning for improved exploration. Their practical performance depends on the quality of the differentiable extension of U , the accuracy of the local or quadratic approximation, and the cost of evaluating reverse proposals in the Metropolis–Hastings correction. The adjusted variants provide asymptotically exact sampling under the usual irreducibility and aperiodicity conditions; unadjusted variants can be cheaper, but generally introduce stationary bias.

Part II

Laplace approximations

Chapter 7

Introduction

Astute readers of the preceding chapter will have noticed that the methods introduced in Part I come with a concerning amount of computational and memory overhead. Even a perfectly efficient Monte Carlo method, one that draws an i.i.d. sample from the posterior in every single step, still requires us to *store* multiple samples—entire copies of the parameter vector—in memory. At prediction time, we then have to compute the forward pass of the model for each of these samples. The number k of samples enters at least multiplicatively into the computational cost c of whatever it is we are doing with our neural network: We will pay a price of $\Omega(kc)$ (the Landau Ω . Yes, that's a *lower* bound), both in memory and computation, to turn a non-Bayesian computation into a Bayesian one. At a time when prominent non-Bayesian deep models are already maxing out the capacity of the best multi-GPU workstations, this is a nonstarter, and arguably the main reason why Bayesian deep learning has acquired a reputation for unacceptably high computational cost.

This Part, however, will introduce a different, more practically minded way to endow neural networks with Bayesian uncertainty. Its history is just as old as that of sampling methods, and was born out of computational necessity in the early days of deep learning [1105]. In fact, it was originally invented in the early 19th century, long before digital computers [974], to turn intractable integrals into tractable ones, at any cost. As computation is once again a bottleneck in the deep learning age, Laplace approximations have re-emerged as a practical tool [380, 812, 884] with new twists that push the envelope of automatic differentiation [365, 366] and numerical linear algebra [309]. It also offers new perspectives on deep learning from the perspective of differential geometry [931, 1427]. As we will see, it now comes with great power, and grave dangers.

7.1 Laplace approximations are local and differential

As in previous chapters, consider a differential program $f : \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$ mapping inputs $x \in \mathbb{R}^D$ to outputs $f(x, \theta) \in \mathbb{R}^C$, parametrised by $\theta \in \mathbb{R}^P$. Consider the regularized empirical risk $\mathcal{L}_\Omega \in \mathbb{R}$ defined as

$$\mathcal{L}_\Omega(\theta) = \sum_{n=1}^N \ell(f(x_n, \theta), y_n) + \Omega(\theta). \quad (7.1)$$

with loss function $\ell : \mathbb{R}^C \times \mathbb{R}^C \rightarrow \mathbb{R}$ and regularizer $\Omega : \mathbb{R}^P \rightarrow \mathbb{R}$, both strictly convex and twice continuously differentiable.¹

Also as in previous chapters, we interpret this function as an energy function, the unnormalised negative logarithm of a posterior:

$$\mathcal{L}_\Omega(\theta) = -\log p(Y | X, \theta) - \log p(\theta) + \text{const} = -\log p(\theta | \mathcal{D}) + \text{const}. \quad (7.2)$$

Definition 7.1. *A Laplace approximation is any algorithm within a general class of methods that approximate $\mathcal{L}_\Omega(\theta)$ with a quadratic function*

$$\mathcal{L}_\Omega(\theta) \approx \frac{1}{2}(\theta - \theta_*)^\top \Psi(\theta_*)(\theta - \theta_*) + \text{const}, \quad (7.3)$$

where $\Psi(\theta_*) \in \mathbb{R}^{P \times P}$ is a second-order linear differential operator acting on $\mathcal{L}_\Omega(\theta_*)$ at the point $\theta_* \in \mathbb{R}^P$. A Laplace approximation is thus defined by the tuple (θ_*, Ψ) , which we will call a **point estimate** and its **curvature**, respectively.

¹Note that convexity here is only assumed for the loss and regularizer. The full loss $\mathcal{L}_\Omega(\theta)$ is not generally convex, since the network f is a nonlinear function of the parameters θ .

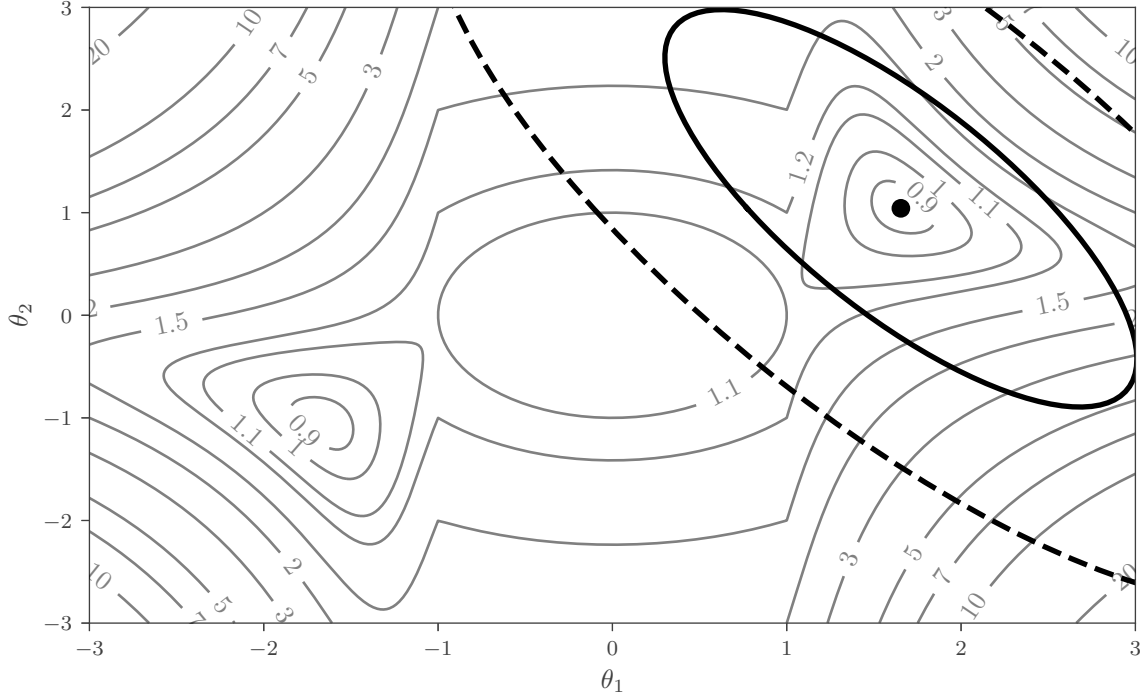


Figure 7.1: A linearized Laplace approximation on the most basic feed-forward perceptron $f(x, \theta) = \theta_2 \cdot \text{ReLU}(\theta_1 x + 1)$ for the dataset $X = (1, -1)$, $Y = (-1, -1)$, with regularizer $\Omega(\theta) = 0.2 \cdot \|\theta\|^2$ and square loss. The grey contour lines denote the energy function. The black dot represents the minimum θ_* ; the thick black ellipses denote one (solid) and two (dashed) standard deviations of the Laplace approximation. The Gaussian approximation defined by the Laplace approximation clearly captures the *local* geometry, but not the full *global* shape of the posterior.

Under the assumption of Equation 7.2, any Laplace approximation is directly associated with an approximate **Gaussian measure on the parameter space**,

$$p(\theta \mid \mathcal{D}) = \frac{1}{Z(\mathcal{D})} \exp(-\mathcal{L}_\Omega(\theta)) \quad (7.4a)$$

$$\approx \frac{1}{Z(\mathcal{D})} \exp\left(-\frac{1}{2}(\theta - \theta_*)^\top \Psi(\theta_*)(\theta - \theta_*)\right) \quad (7.4b)$$

$$= \mathcal{N}(\theta; \theta_*, \Psi^{-1}(\theta_*)). \quad (7.4c)$$

The textbook, but impractical, version of the Laplace approximation arises from the Taylor expansion at a local minimizer of the regularized risk:

$$[\Psi(\theta)]_{ij} = [\nabla_\theta \nabla_\theta^\top \mathcal{L}_\Omega(\theta)]_{ij} = \frac{\partial^2 \mathcal{L}_\Omega(\theta)}{\partial \theta_i \partial \theta_j}, \quad \theta_* = \arg \min_{\theta \in \mathbb{R}^P} \mathcal{L}_\Omega(\theta). \quad (7.5)$$

That is, we pick a point θ_* where the gradient vanishes, and compute the Hessian there (the matrix $\Psi(\theta_*)$ being the matrix representation of the aforementioned operator, evaluated at θ_*).

7.2 What is to like about Laplace?

Central to the appeal of Laplace approximations is that they are an *analytic post-processing* step that can be tagged onto the established way of training a deep model:

How to construct a Laplace approximation:

1. Train your deep network in any way you know and trust, using whatever optimiser you like. Wait for your training loop to terminate and return the point estimate θ_* .
2. After training has finished (!), compute $\Psi(\theta_*)$. Since it is a second-order differential operator, it can be computed with automatic differentiation, using your favourite auto-diff framework.

And that's it! You now have a “Bayesian uncertainty” estimate in parameter space, in the form of the Gaussian from Equation 7.4c. Let us spell out the advantages of this recipe, some more obvious, some less so, explicitly:

- **You get to keep your point-estimate.** People who train neural networks, especially if it is a large model, spend extreme resources to find a good value θ_* . Even if this particular value may not hold up to the theoretical demands of Bayesian analysis, it nevertheless embodies a significant investment, so it is appealing to keep working with this valuable object. For Laplace, this point estimate turns directly into a posterior mean. And to state the obvious: The computation overhead of this part is strictly *zero* over whatever the training cost to begin with.
- **You get to use established tools.** Constructing the Gaussian of Equation 7.4c, requires precisely two computational tools: *Automatic differentiation* to get Ψ , and *numerical linear algebra*, to find its (approximate) inverse. These are arguably *the* most powerful, reliable, optimised functionalities available to the machine learning engineer. They can be made extremely numerically robust and conveniently automated, or tailored for radical computational efficiency. And, using probabilistic numerical linear algebra, the discretisation and iteration error of these internal computations can be tracked for downstream consideration (More about numerical linear algebra tricks in Chapter 8 and Part VIII).
- **Laplace is post hoc.** Since Ψ is only needed after θ_* has been found, its cost is not a function of the complexity of training. In fact, the construction of a Laplace approximation is effectively separated from training entirely, and can be done on a pre-trained model downloaded from the Internet. The only caveat is that, since Ψ is a function of the regularized risk, we need access to the training data $\mathcal{D}$ – or some compressed representation thereof.
- **Laplace is analytical.** Laplace approximations return a Gaussian probability measure, whose parameters are computed using optimization, automatic differentiation, and linear algebra. That is, a deterministic quantity – a probability density function – constructed deterministically. There is no source of randomness involved, apart from the potentially stochastic nature of training to find θ_* . As we will see in Chapter 10, this makes Laplace approximations interesting for downstream applications and decisions that require analytic marginalization over the parameters.
- **Deep theoretical meaning.** We will find in Chapters 8 and 9 that the local differential nature of the curvature at the heart of Laplace approximations gives them profound theoretical connections, and intuitive interpretations. In fact, they open up a whole self-contained theory of deep learning founded on the differential geometry of the energy manifold associated with $\mathcal{L}_\Omega$.
- **Laplace turns every deep network into a Gaussian process.** In Chapter 8, we will see that the Gaussian measure $\mathcal{N}(\theta; \theta_*, \Psi^{-1}(\theta_*))$ constructed by the Laplace approximation can be analytically pushed forward to the output space of the network $f(x, \theta)$, where it manifests itself as a Gaussian process, whose mean-function is exactly the trained network $f(x, \theta_*)$, and whose covariance function (kernel) is given by a structured object, a *tangent kernel* (developed further in Part V). This is a powerful perspective for both practical and conceptual reasons: On the practical side, we can more or less directly transport a lot of notions from the Gaussian process literature to deep learning, recovering advanced functionality sorely missing from non-Bayesian deep learning. And on the conceptual side, the tangent kernel contains very specific terms that capture modular, interpretable statements about the role of the training data, the network architecture, and regularization that make deep learning more transparent and interpretable.

7.3 Practical Considerations

On first encountering Laplace approximations in deep learning, people tend to have one of two, mutually almost contradictory reactions:

7.3.1 Isn't this too simplistic?

A Taylor-expansion? Really? From a fundamentalist Bayesian perspective, a quadratic function is a poor approximation of the posterior, with all its non-convex nonlinear glory (cf. Figure 7.1). But compared to the day-to-day practice of training deep networks, Laplace approximations are a big step forward, from a *zeroth order* approximation (a point estimate) to a *second-order* approximation (a Gaussian probability measure). They offer an easy gateway into the Bayesian world for the deep learning practitioner: Just take your trained model, compute a curvature, and Bob's your uncle.

We should also be careful not to appeal to unrealistic ideals, but compare to realistic alternatives instead. The true posterior may be the theoretical target of all Bayesian analysis, but is completely intractable. All Bayesian methods that are actually accessible are approximations, of different nature. For example, if we draw an ensemble of $S \ll P$ samples $[\theta_s]_{s=1}^S$ from the posterior, and use it to construct a posterior covariance $\hat{\Psi}^{-1} = \frac{1}{S} \sum_{s=1}^S (\theta_s - \theta_*)(\theta_s - \theta_*)^\top$ over the parameters θ , that matrix $\hat{\Psi}$ will be of rank at most S . If we use this object as a proxy for “posterior uncertainty”, we may severely over-estimate confidence intervals. A Laplace approximation might overlook nonlocal aspects of the posterior's geometry, but it can construct an approximate covariance matrix of full rank.

7.3.2 Isn't this too expensive?

Hang on, we need a Hessian over the parameter space? That's a matrix of size $P \times P$, so just storing it will create a memory overhead of $\mathcal{O}(P^2)$. Oh, and then we need to compute its inverse!? That will incur a computational cost of $\mathcal{O}(P^3)$, right? The flaw of such thoughts is that they use worst case statements about objects that are entirely under the user's control.

It is better to think of the curvature Ψ as a *linear operator* [366], a function mapping vectors to vectors. Such a function need not require the explicit instantiation of a matrix in memory. Chapters 8 and 10 will introduce several possible choices for this operator, spanning a range of computational and memory costs. These include the generalised Gauss-Newton approximation (Section 8.3), Kronecker-factored approximate curvature, and diagonal approximations to the Hessian (Section 8.4). In fact, in Section 10.1 we will see that even the trivial choice $\Psi = \alpha I$ for an arbitrary $\alpha \in \mathbb{R}_+$ (i.e. just replacing the point estimate with an arbitrary Gaussian measure) already fixes some tangible, nontrivial pathologies of non-Bayesian deep learning.

Likewise, numerical linear algebra offers a rich toolbox of iterative, sparse, sketched, etc. methods to solve linear systems of equations of the form $\Psi x = b$ for x without the need to even explicitly form an estimate of the matrix inverse Ψ^{-1} (Section 8.4.1). And there are probabilistic versions of these methods that can partially track and estimate the error of their output [e.g. 317, 1740, 1741]. The memory and compute cost of Laplace approximations can thus be completely controlled by the user, opening up a wide spectrum of algorithms of varying quality and cost.

On the flip side, this also means that constructing Laplace approximations, just like training deep networks, is not a turnkey operation, and requires care and expertise. But emerging software tools increasingly encapsulate these choices, and provide abstraction layers that allow constructing Laplace approximations with a few lines of code.

7.4 All approximations are wrong

Despite all the arguments in their favour, Laplace approximations remain just that: approximations. Their local nature – the fact that they only use differential quantities evaluated at a single point in the parameter space – is both their principal strength and their principal weakness. As Figure 7.1 shows, the Gaussian posterior created by a Laplace approximation may not capture the geometry of the posterior – not even within the region of the parameter space delineated by one or two standard deviations of that Gaussian. And when intricate numerical routines are used to operate on the curvature operator Ψ , their own errors can also severely erode the quality of the approximation.

Laplace approximations should thus best be viewed as an important tool of first resort. At least their most basic variants can be constructed on virtually any deep architecture, often with minimal overhead in both implementation and computation. They can be a powerful diagnostic tool, elucidating the interplay between data and architecture; and offer structured, controllable functionality for everyday requirements like uncertainty quantification, model adaptation, continual learning, and more. But whenever the stakes are high, for example in safety critical applications, or when uncertainty is used to draw scientific conclusions, additional safeguards are needed. In such settings, Laplace approximations may be paired with some of the other frameworks described in this book,

to provide a more robust and calibrated uncertainty estimate.

Takeaways:

A Laplace approximation is a quadratic approximation to the log posterior associated with a deep model, interpreted as a Gaussian posterior on the parameter space, or a Gaussian process posterior on the predictive (logit) outputs of the model. Such approximations are constructed by first training a point estimate for the parameters in the “normal” way, then computing the curvature operator (or kernel) post hoc. By this nature, Laplace approximations are *local* and *differential*. On the one hand, this is a weakness, because they fail to capture *global, non-linear* structure in the posterior. On the other hand, this is a strength, because it allows the use of powerful automatic differentiation and matrix-free, probabilistic linear algebra, and provides theoretical descriptions of deep learning in the terms of differential geometry. The following chapters will provide further details on these aspects.

Chapter 8

Linearised Laplace approximations

The previous chapter showed a computationally lightweight Gaussian approximation $\mathcal{N}(\theta; \theta_*, \Psi(\theta_*)^{-1})$ to the true posterior $p(\theta \mid \mathcal{D})$ on the parameter space $\mathbb{R}^P$. Our final goal is to use this Gaussian measure to *make decisions* based on the model’s output: its predictions feed into the risks and utilities that downstream tasks optimise (Chapter 10). To this end, we first quantify the **predictive uncertainty** of our network on the output space $\mathbb{R}^C$ by turning it into a Gaussian process (GP). Figure 8.1 previews the pipeline, from the maximum a posteriori (MAP) estimate θ_* to the predictive distribution.

8.1 Linearisation

To turn an arbitrary (differentiable) neural network parameterised by $\theta \in \mathbb{R}^P$ into a Gaussian process, we use two ingredients: linearisation in the parameter space and the previously introduced Laplace approximation.

We first introduce the concept of the **linearised network** $f^{\theta_0}: \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$ as the first-order Taylor approximation of network f around parameters $\theta_0 \in \mathbb{R}^P$:

$$f^{\theta_0}(x, \theta) := f(x, \theta_0) + J_\theta f(x, \theta_0)(\theta - \theta_0), \quad (8.1)$$

where $J_\theta f(x, \theta_0) \in \mathbb{R}^{C \times P}$ is the **network Jacobian** evaluated at (x, θ_0) .¹ Analogously to Equation 7.1, we define the regularised empirical risk of the linearised network f^{θ_0} with parameters θ as

$$\mathcal{L}_\Omega^{\theta_0}(\theta) := \sum_{n=1}^N \ell(f^{\theta_0}(x_n, \theta), y_n) + \Omega(\theta). \quad (8.2)$$

Unless explicitly stated otherwise (e.g., in the prior-only discussion in Section 8.2), we fix the linearisation point at a minimiser of the *original* regularised empirical risk $\mathcal{L}_\Omega(\theta)$:

$$\theta_* := \arg \min_{\theta \in \mathbb{R}^P} \mathcal{L}_\Omega(\theta), \quad (8.3)$$

that is, we set $\theta_0 \equiv \theta_*$. Linearisation itself only requires differentiability of f at θ_0 . The Laplace approximation needs more: θ_* must be a local minimiser of $\mathcal{L}_\Omega$ with zero gradient, and the resulting precision must be positive definite after adding the prior curvature. We note that f^{θ_*} shares its value and its parameter Jacobian with f at θ_* , so $\nabla_\theta \mathcal{L}_\Omega^{\theta_*}(\theta_*) = \nabla_\theta \mathcal{L}_\Omega(\theta_*) = 0$ and θ_* is a stationary point of $\mathcal{L}_\Omega^{\theta_*}$, too. Because f^{θ_*} is affine in θ , convexity of $\ell(\cdot, y)$ in z makes each loss term convex in θ ; with a strictly convex Ω , the risk $\mathcal{L}_\Omega^{\theta_*}$ is then strictly convex in θ , hence θ_* is its unique minimiser.

Following Section 7.1, the Laplace approximation using the regularised empirical risk of the linearised network yields the parameter-space Gaussian measure $\mathcal{N}(\theta; \theta_*, \Psi(\theta_*)^{-1})$ with

$$[\Psi(\theta)]_{ij} = \left[\nabla_\theta \nabla_\theta^\top \mathcal{L}_\Omega^{\theta_*}(\theta) \right]_{ij} = \frac{\partial^2 \mathcal{L}_\Omega^{\theta_*}(\theta)}{\partial \theta_i \partial \theta_j}. \quad (8.4)$$

The curvature Ψ of the linearised network’s regularised empirical risk differs from the Hessian in Equation 7.5 for networks that are not piecewise affine in their parameters (except at the non-generic parameter values where

¹Note that this network is only linear in its parameters θ but is still nonlinear in its inputs x .

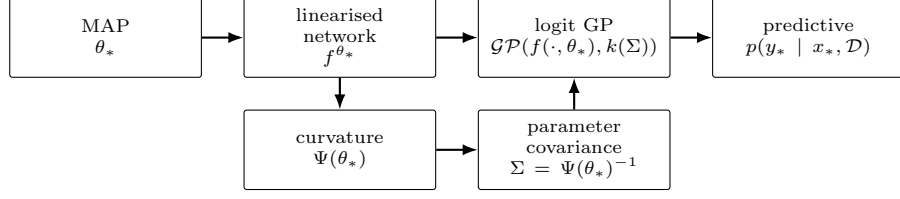


Figure 8.1: The linearised Laplace pipeline. The linearised network f^{θ_*} and parameter covariance $\Sigma = \Psi(\theta_*)^{-1}$ jointly determine the logit Gaussian process, which induces a predictive distribution.

all training residuals vanish). The difference is that $\Psi(\theta_*)$, as defined in Equation 8.4, discards the network's parameter curvature (and higher-order derivatives) and only retains the curvature of the *loss function* over the model outputs. The advantages of this modelling choice are summarised in Section 8.1.1.

An important property of Gaussian measures is their closure under affine transformations, stated formally below.

Proposition 8.1 (Affine Closure of Gaussians). *Let $X \sim \mathcal{N}(\mu, \Sigma)$ with $\mu \in \mathbb{R}^N$ and $\Sigma \succ 0$. For any $A \in \mathbb{R}^{M \times N}$ with full row rank and $b \in \mathbb{R}^M$,*

$$Y := AX + b \sim \mathcal{N}(A\mu + b, A\Sigma A^\top). \quad (8.5)$$

If $\text{rank}(A) < M$, then $A\Sigma A^\top \succeq 0$ and Y has no Lebesgue density on $\mathbb{R}^M$.

Proof. We first treat an invertible $A \in \mathbb{R}^{N \times N}$. For any Borel set $E \subseteq \mathbb{R}^N$, the event $\{Y \in E\}$ equals $\{X \in A^{-1}(E - b)\}$, so substituting $x = A^{-1}(y - b)$, whose Jacobian has absolute determinant $|\det A^{-1}|$, gives

$$p_Y(y) = p_X(A^{-1}(y - b)) |\det A^{-1}|. \quad (8.6)$$

Writing $d := y - (A\mu + b)$, we have $A^{-1}(y - b) - \mu = A^{-1}d$, so

$$(A^{-1}(y - b) - \mu)^\top \Sigma^{-1} (A^{-1}(y - b) - \mu) = d^\top (A\Sigma A^\top)^{-1} d, \quad (8.7)$$

and $|\Sigma|^{-1/2} |\det A^{-1}| = |A\Sigma A^\top|^{-1/2}$ by $|A\Sigma A^\top| = |A|^2 |\Sigma|$. Substituting the Gaussian density of X yields

$$p_Y(y) = (2\pi)^{-N/2} |A\Sigma A^\top|^{-1/2} \exp\left(-\frac{1}{2} d^\top (A\Sigma A^\top)^{-1} d\right), \quad (8.8)$$

so $Y \sim \mathcal{N}(A\mu + b, A\Sigma A^\top)$.

For the full row rank case $A \in \mathbb{R}^{M \times N}$ with $M \leq N$, choose $K \in \mathbb{R}^{(N-M) \times N}$ so that $T := \begin{bmatrix} A \\ K \end{bmatrix} \in \mathbb{R}^{N \times N}$ is invertible, and set $Y := AX + b$, $Z := KX$, and $U := \begin{bmatrix} Y \\ Z \end{bmatrix} = TX + \begin{bmatrix} b \\ 0 \end{bmatrix}$. By the invertible case, U is Gaussian with mean $\begin{bmatrix} A\mu + b \\ K\mu \end{bmatrix}$ and covariance

$$S := T\Sigma T^\top = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} = \begin{bmatrix} A\Sigma A^\top & A\Sigma K^\top \\ K\Sigma A^\top & K\Sigma K^\top \end{bmatrix}, \quad S_{11} = A\Sigma A^\top \succ 0. \quad (8.9)$$

Let $C := S_{22} - S_{21}S_{11}^{-1}S_{12}$ be the (positive definite) Schur complement and $B := S_{21}S_{11}^{-1}$. With $\Delta y := y - (A\mu + b)$ and $\Delta z := z - K\mu$, the block-inverse identity splits the quadratic form as

$$\begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix}^\top S^{-1} \begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix} = \Delta y^\top S_{11}^{-1} \Delta y + (\Delta z - B\Delta y)^\top C^{-1} (\Delta z - B\Delta y), \quad (8.10)$$

and $|S| = |S_{11}||C|$. Marginalising U over z with

$$\int_{\mathbb{R}^{N-M}} \exp\left(-\frac{1}{2} (\Delta z - B\Delta y)^\top C^{-1} (\Delta z - B\Delta y)\right) dz = (2\pi)^{(N-M)/2} |C|^{1/2} \quad (8.11)$$

leaves

$$p_Y(y) = (2\pi)^{-M/2} |S_{11}|^{-1/2} \exp\left(-\frac{1}{2} \Delta y^\top S_{11}^{-1} \Delta y\right), \quad (8.12)$$

that is, $Y \sim \mathcal{N}(A\mu + b, A\Sigma A^\top)$. Only $S_{11} = A\Sigma A^\top$ survives, so the result is independent of the choice of K .

Finally, if $\text{rank}(A) < M$, then $\text{rank}(A\Sigma A^\top) = \text{rank}(A) < M$ because $\Sigma \succ 0$, so $A\Sigma A^\top$ is singular and Y is supported on the proper affine subspace $A\mathbb{R}^N + b$, hence has no Lebesgue density on $\mathbb{R}^M$. $\square$

As such, the linearised network f^{θ_*} pushes the parameter posterior forward, via the affine map of Equation 8.1, to a Gaussian in the output space (the logit space in classification). In the network-output variable $z \in \mathbb{R}^C$ (the logits, which the loss $\ell(z, y)$ takes as its first argument), this measure is

$$\mathcal{N}(z; f(x_*, \theta_*), J_\theta f(x_*, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x_*, \theta_*)^\top). \quad (8.13)$$

This result already equips the network f with a probability measure in its output space (or logit space in classification) for every input x_* . A **Gaussian process** is an even more powerful construction, capable of modelling a Gaussian measure over any *finite set of inputs*, not just a single one. The linearised network f^{θ_*} equipped with its parameter-space Laplace approximation defines a Gaussian process $\mathcal{GP}(f(\cdot, \theta_*), k)$ with the following (Mercer) kernel:

$$k(x, x') = \text{cov}(f^{\theta_*}(x, \theta), f^{\theta_*}(x', \theta)) \quad (8.14)$$

$$= \text{cov}(J_\theta f(x, \theta_*)(\theta - \theta_*), J_\theta f(x', \theta_*)(\theta - \theta_*)) \quad (8.15)$$

$$= J_\theta f(x, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x', \theta_*)^\top. \quad (8.16)$$

8.1.1 Advantages of Using the Linearised Network

We perform the Laplace approximation w.r.t. $\mathcal{L}_\Omega^{\theta_*}$ (with $\theta_* = \arg \min_{\theta \in \mathbb{R}^P} \mathcal{L}_\Omega(\theta)$) instead of the regularised empirical risk of f for multiple reasons.

First, the Hessian of the regularised empirical risk of f (Equation 7.5) is not guaranteed to be positive definite, preventing us from interpreting it as the inverse covariance matrix of our approximate Gaussian posterior over the parameters θ .² In second-order optimisation, the lack of positive definiteness also prohibits using the inverse Hessian as a preconditioner. However, the Hessian of the regularised empirical risk of f^{θ_*} , with a strongly convex regulariser, is *guaranteed* to be positive definite.

Second, the Hessian of the regularised empirical risk of f^{θ_*} is considerably cheaper to use than that of f , allowing Laplace approximations to scale to deep learning architectures (see Section 8.3).

There are thus two uses of the linearised model. First, it defines the curvature Ψ and hence the parameter-space Gaussian posterior approximation. Second, it defines the predictive pushforward in Equation 8.13. Earlier work on Laplace approximations for neural networks [1108, 1402] *sampled* M parameter configurations $\{\theta_s\}_{s=1}^M$ from the parameter-space Gaussian and calculated *predictive* quantities from the nonlinear predictions $\{f(x, \theta_s)\}_{s=1}^M$. This loses the exact output-space Gaussian distribution available from linearisation and yields no posterior GP. Lawrence [984] shows empirically that this approach is prone to underfitting: the predictive becomes *too uncertain* even on in-distribution data points. Immer et al. [813] identify the core reason for performance loss as a mismatch between the model in which we perform approximate Bayesian inference (the linearised network f^{θ_*}) and the model we use for prediction (the nonlinear network f). Figure 8.2 visualises this underfitting and the low-density samples drawn under the true posterior. See also Immer et al. [813, Fig. 3]. By using the linearised network to make predictions and not only to calculate the curvature, the approximation remains internally consistent.

Takeaways

The linearised Laplace approximation turns any differentiable neural network into a *Gaussian process* by (i) linearising the network in parameter space at θ_* and (ii) placing a Gaussian posterior $\mathcal{N}(\theta_*, \Psi(\theta_*)^{-1})$ on parameters. The induced Mercer kernel has the form

$$k(x, x') = J_\theta f(x, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x', \theta_*)^\top,$$

whereby Ψ captures the effect of the dataset on the *parameter-space curvature*, which the Jacobian $J_\theta f(\cdot, \theta_*)$ pushes forward into *function-space uncertainty*.

²In particular, the transition from Equation 7.4b to Equation 7.4c is not justified when $\Psi(\theta_*)$ is not positive definite.

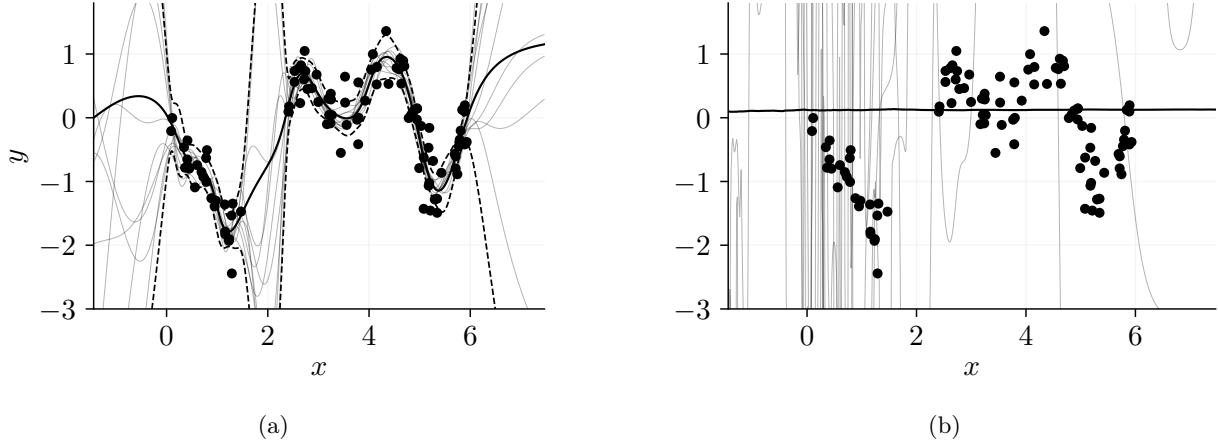


Figure 8.2: The same full Laplace posterior under two predictive rules on a one-dimensional regression example from the code accompanying Immer et al. [813]. Both panels use the same parameter-space approximate posterior and the same ten parameter draws. They differ only in how each parameter draw becomes a function-space sample called a *sample path* in the GP literature. Panel (a) sends each draw through the linearised network $f^{\theta*}$, the model consistent with the parameter-space Laplace approximation. Panel (b) sends the draws through the original nonlinear network f . Solid curves are predictive means; dashed curves show pointwise 95% credible intervals. The linearised Laplace approximation in Panel (a) yields a GP; therefore, the output-space quantities are available in closed form. The (nonlinear) Laplace approximation in Panel (b) yields an intractable output mean and covariance, which we approximate with two million Monte Carlo samples. The nonlinear scheme is far more uncertain over the training inputs and its mean severely underfits. The linearised Laplace approximation yields consistent function-space uncertainty.

8.2 The Prior

The Gaussian process determined by the *posterior* parameter-space covariance matrix Ψ^{-1} is tied to the dataset $\mathcal{D}$, as Ψ is the Hessian of the linearised model’s regularised empirical risk over $\mathcal{D}$. It is worth spending some time on the case when the neural network $f: \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$ has not been exposed to any dataset $\mathcal{D}$ yet. In this case, the model’s behaviour is determined by its architecture and by the prior $p(\theta)$ over the parameters.

For a non-Gaussian parameter prior $p(\theta)$, a Laplace approximation to the prior is centred at a local maximum θ_0 of $p(\theta)$, analogously to Equation 7.4c. Its precision is the negative log-prior Hessian,

$$\Psi = -\nabla_{\theta} \nabla_{\theta} \log p(\theta_0). \quad (8.17)$$

For the construction that follows, we specialise to a Gaussian prior centred at the linearisation point,

$$p(\theta) = \mathcal{N}(\theta; \theta_0, I), \quad (8.18)$$

where θ_0 can be any chosen initialisation. The prior’s mode is θ_0 , so the linearisation and the Laplace approximation of the prior share the same centre, and the precision above evaluates to $\Psi = I$. Pushing this prior through the linearised network gives the kernel $k: \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}^{C \times C}$,

$$k(x, x') = J_{\theta} f(x, \theta_0) (J_{\theta} f(x', \theta_0))^{\top}. \quad (8.19)$$

The kernel comes from the same pushforward as the posterior GP kernel in Equation 8.16. Compared with it, the data-dependent pair $(\theta_*, \Psi(\theta_*))$ is replaced by the dataset-independent pair (θ_0, I) . This kernel is known as the **Neural Tangent Kernel (NTK)** [829].³ It differs from the neural network Gaussian process (NNGP) kernel: the NNGP kernel is the covariance of random network outputs $f(x, \theta)$ and $f(x', \theta)$ under a random initialisation distribution, while the NTK is the covariance of the linearised network outputs $f^{\theta_0}(x, \theta)$ and $f^{\theta_0}(x', \theta)$ under the prior $\mathcal{N}(\theta; \theta_0, I)$. As the network width is taken to infinity in a specific way, both kernels are convergent. Where these limits exist, they are different deterministic kernels.

³For a finite-width network, this is sometimes called the *empirical* NTK at θ_0 , and the NTK is reserved for networks in the infinite-width limit.

The NTK determines the function-space trajectory of f during full-batch gradient flow training on the average empirical loss via the Ordinary Differential Equation (ODE)

$$\frac{d}{dt}f_t(x) = -\frac{1}{N} \sum_{n=1}^N k_t(x, x_n) \nabla_z \ell(z, y_n)|_{z=f_t(x_n)}, \quad (8.20)$$

where t is the time variable of the gradient flow and k_t is the NTK of f_t . Jacot et al. [829] show that in the infinite-width limit of fully-connected neural networks with the so-called NTK initialisation, the NTK stays constant during training, and the convergence of optimisation is related to the positive definiteness of the NTK. However, the NTK of regular, finite-width neural networks is not constant and must be recomputed at every evaluation point.

From a modelling perspective, putting an isotropic Gaussian prior over the parameters is seldom well-motivated. However, the framework of Laplace approximations is compatible with interpretable function-space priors. Cinquin et al. [309] define Laplace priors using a prior Gaussian process. In particular, the negative log-prior is, up to an additive constant, equal to the squared Reproducing Kernel Hilbert Space (RKHS) norm of the difference between the linearised network f^{θ_0} and the mean of the prior Gaussian process μ . This negative log-prior is also used as a regulariser during training. As specifying the Gaussian prior in the function space only requires knowledge about the function's behaviour and *not* about the parameter space's structure, the framework of Cinquin et al. [309] allows domain experts to directly encode functional behaviour in the Bayesian inference problem.

Takeaways

Without any data and with a Gaussian prior $\mathcal{N}(\theta; \theta_0, I)$ centred at θ_0 , the parameter-space curvature is the identity, $\Psi = I$, and the GP kernel derived from neural networks reduces to the Neural Tangent Kernel (NTK):

$$k(x, x') = J_\theta f(x, \theta_0) J_\theta f(x', \theta_0)^\top.$$

8.3 The Fisher

In Section 8.1, we glossed over the curvature matrix defined in Equation 8.4. This matrix and its components give an information-geometric interpretation to Laplace approximations, warranting its own chapter (Chapter 9).

Consider the regularised empirical risk of the linearised network f^{θ_0} (Equations 8.1 and 8.2). Under the assumption of Equation 7.2 applied to the linearised network, the term

$$\mathcal{L}^{\theta_0}(\theta) = \sum_{n=1}^N \ell(f^{\theta_0}(x_n, \theta), y_n) \quad (8.21)$$

is equal, up to an additive constant, to the negative log-likelihood of the dataset under the model, $-\log p(Y | X, \theta)$, whereas the term $\Omega(\theta)$ equals the negative log-prior, $-\log p(\theta)$, again up to an additive constant.

Writing out the Hessian of $\mathcal{L}^{\theta_0}$ evaluated at θ_0 , we obtain

$$G(\theta_0) = \nabla_\theta \nabla_\theta^\top \mathcal{L}^{\theta_0}(\theta_0) = \sum_{n=1}^N (J_\theta f(x_n, \theta_0))^\top \left(\nabla_z \nabla_z^\top \ell(z, y_n) \Big|_{z=f^{\theta_0}(x_n, \theta_0)} \right) J_\theta f(x_n, \theta_0), \quad (8.22)$$

called the **Generalised Gauss-Newton** matrix (GGN). In this section, we show that this matrix is equivalent to the Fisher Information Matrix for a family of loss functions, which is a natural (usually degenerate) Riemannian metric for statistical manifolds parameterised by the model (Section 9.2).

Consider the statistical manifold

$$\mathcal{M} = \{p(x, y | \theta) : \theta \in \mathbb{R}^P\} \quad (8.23)$$

with conditional independence structure $p(x, y | \theta) = p(y | x, \theta) \cdot p(x)$, whose in-depth study we defer to Chapter 9. The $p(y | x, \theta)$ distribution represents the likelihood parameterised by a neural network and $p(x)$ is the ground-truth input distribution. A natural notion of separation between probability distributions on this manifold is the **Kullback-Leibler (KL) divergence**:

$$D_{\text{KL}}(p(x, y | \theta_1) \parallel p(x, y | \theta_2)) := \mathbb{E}_{p(x, y | \theta_1)} \left[\log \frac{p(x, y | \theta_1)}{p(x, y | \theta_2)} \right]. \quad (8.24)$$

The KL divergence is not a valid metric, as it is non-symmetric. However, for parameters θ_1 and θ_2 and $\Delta\theta = \theta_1 - \theta_2$, the KL divergence's second-order Taylor approximation around θ_1 is given by

$$D_{\text{KL}}(p(x, y | \theta_1) \parallel p(x, y | \theta_2)) = \frac{1}{2} \Delta\theta^\top F(\theta_1) \Delta\theta + \mathcal{O}(\|\Delta\theta\|^3). \quad (8.25)$$

Both the zeroth- and first-order terms vanish (the former because $D_{\text{KL}}(p \parallel p) = 0$, the latter because the expected score is zero, Equation 8.27), leaving the quadratic Fisher term as the leading contribution. At each parameter θ , the bilinear form $g_\theta(u, v) := u^\top F(\theta)v$ on tangent vectors u, v is a valid Riemannian metric over manifold $\mathcal{M}$ (up to the identifiability of parameters; see Chapter 9). The matrix

$$F(\theta) = \mathbb{E}_{p(x, y | \theta)} [-\nabla_\theta \nabla_\theta^\top \log p(x, y | \theta)] \quad (8.26)$$

is called the **Fisher Information Matrix (FIM)** of the random variable (X, Y) distributed according to $p(x, y | \theta)$.

Differentiating the normalisation $\int p(x, y | \theta) dx dy = 1$ under the integral sign and using $\nabla_\theta p = p \nabla_\theta \log p$ gives the zero-mean score,

$$\mathbb{E}_{p(x, y | \theta)} [\nabla_\theta \log p(x, y | \theta)] = 0. \quad (8.27)$$

Differentiating once more yields the two equivalent forms of the Fisher information,

$$F(\theta) = \mathbb{E}_{p(x, y | \theta)} [-\nabla_\theta \nabla_\theta^\top \log p(x, y | \theta)] = \mathbb{E}_{p(x, y | \theta)} [\nabla_\theta \log p(x, y | \theta) (\nabla_\theta \log p(x, y | \theta))^\top]. \quad (8.28)$$

The FIM of independent random variables is additive. In particular, for N i.i.d. random variables $\{(X_n, Y_n)\}_{n=1}^N$, the FIM is N times that of a single random variable (X, Y) :

$$\mathbb{E}_{p(\{(x_n, y_n)\}_{n=1}^N | \theta)} [-\nabla_\theta \nabla_\theta^\top \log p(\{(x_n, y_n)\}_{n=1}^N | \theta)] \quad (8.29)$$

$$= \sum_{n=1}^N \mathbb{E}_{p(x_n, y_n | \theta)} [-\nabla_\theta \nabla_\theta^\top \log p(x_n, y_n | \theta)] \quad (8.30)$$

$$= NF(\theta). \quad (8.31)$$

In the following, we use the notation $F(\theta)$ for the FIM of a single random variable (X, Y) distributed according to $p(x, y | \theta)$.

Using Equation 8.25, the utility of the FIM becomes clear in second-order optimisation. In particular, consider the objective

$$\arg \min_{\Delta\theta \in \mathbb{R}^P} \mathcal{L}(\theta + \Delta\theta) \quad \text{s.t.} \quad D_{\text{KL}}(p(x, y | \theta) \parallel p(x, y | \theta + \Delta\theta)) \leq c \quad (8.32)$$

for some constant $c \in \mathbb{R}_{>0}$. The intuition behind this objective is that we want to minimise our model's non-regularised empirical risk while limiting how much the likelihood induced by the model changes, irrespective of the curvature of the loss surface. To find an approximate minimiser of the objective in Equation 8.32, we consider the quadratic approximation to our constraint in Equation 8.25 and linearise the empirical risk around the current parameter θ :

$$\mathcal{L}(\theta + \Delta\theta) = \mathcal{L}(\theta) + \Delta\theta^\top \nabla_\theta \mathcal{L}(\theta) + \mathcal{O}(\|\Delta\theta\|^2). \quad (8.33)$$

Substituting in the linearised empirical risk and quadratic constraint, our objective becomes

$$\arg \min_{\Delta\theta \in \mathbb{R}^P} \Delta\theta^\top \nabla_\theta \mathcal{L}(\theta) \quad \text{s.t.} \quad \frac{1}{2} \Delta\theta^\top F(\theta) \Delta\theta \leq c. \quad (8.34)$$

The above is a convex quadratically constrained linear program. When $F(\theta)$ is positive definite and $\nabla_\theta \mathcal{L}(\theta) \neq 0$, the optimum lies on the boundary of the constraint ellipsoid and can be found in closed form with Lagrange's method, yielding

$$\Delta\theta^* = -\frac{1}{\lambda^*} F(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta), \quad \lambda^* = \sqrt{\frac{(\nabla_\theta \mathcal{L}(\theta))^\top F(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta)}{2c}}. \quad (8.35)$$

In particular, the optimal update direction $\Delta\theta^*$ is given by the so-called **natural gradient** $F(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta)$ with a magnitude that is a function of the constant c . Natural-gradient methods are developed in detail in Chapter 15.

In practice, we replace the unknown distribution of inputs $p(x)$ with the empirical distribution w.r.t. training set $\mathcal{D}$:

$$p(x) \approx p_{\mathcal{D}}(x) = \frac{1}{N} \sum_{n=1}^N \delta(x - x_n). \quad (8.36)$$

The corresponding FIM approximation becomes

$$\tilde{F}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathbb{E}_{p(y|x_n, \theta)} \left[-\nabla_{\theta} \nabla_{\theta}^{\top} \log p(y | x_n, \theta) \right]. \quad (8.37)$$

Using the chain rule w.r.t. the model's output $z = f(x_n, \theta)$, we obtain

$$\tilde{F}(\theta_0) = \frac{1}{N} \sum_{n=1}^N (J_{\theta} f(x_n, \theta_0))^{\top} \mathbb{E}_{p(y|x_n, \theta_0)} \left[-\nabla_z \nabla_z^{\top} \log p(y | z) \Big|_{z=f(x_n, \theta_0)} \right] J_{\theta} f(x_n, \theta_0). \quad (8.38)$$

Here the network's own second-order term vanishes under the expectation by the zero-mean score in z (Equation 8.27 applied to $p(y | z)$), which is why the linearised and nonlinear networks share the same $\tilde{F}(\theta_0)$. The expectation over model outputs for each input can be expensive to compute. One approach to mitigate the cost is the Monte Carlo approximation:

$$\mathbb{E}_{p(y|x_n, \theta)} \left[-\nabla_z \nabla_z^{\top} \log p(y | z) \Big|_{z=f(x_n, \theta)} \right] \approx -\frac{1}{M} \sum_{s=1}^M \nabla_z \nabla_z^{\top} \log p(y_s | z) \Big|_{z=f(x_n, \theta)}, \quad (8.39)$$

with $y_s \sim p(y | x_n, \theta)$. For small M , we obtain a rather crude but cheap approximation. An alternative approach is to first rewrite $\tilde{F}(\theta)$ as

$$\tilde{F}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathbb{E}_{p(y|x_n, \theta)} \left[\nabla_{\theta} \log p(y | x_n, \theta) (\nabla_{\theta} \log p(y | x_n, \theta))^{\top} \right] \quad (8.40)$$

(Equation 8.28, applied to $p(y | x_n, \theta)$), then substitute the ground-truth labels instead of approximating the expectation:

$$\hat{F}(\theta) = \frac{1}{N} \sum_{n=1}^N \nabla_{\theta} \log p(y_n | x_n, \theta) (\nabla_{\theta} \log p(y_n | x_n, \theta))^{\top}, \quad (8.41)$$

called the **Empirical Fisher Information Matrix (EFIM)**. While this substitution may seem natural, Kunstner et al. [948] show that it has a detrimental effect for optimisation, where the predicted and ground-truth labels can be significantly different, especially at the beginning of training. However, for the Laplace approximation, we use a MAP estimate, which, if it generalises well, approximates the ground-truth label distribution well.

For a wide family of likelihoods, one does *not* have to approximate the Fisher $\tilde{F}(\theta)$ by Monte Carlo. This allows us to connect the FIM to the GGN under some assumptions. Identifying $\ell(z, y_n) = -\log p(y_n | z)$ and $f(x_n, \theta_0) = f^{\theta_0}(x_n, \theta_0)$ using Equation 8.1, in the case when the Hessian of the negative log-likelihood loss ℓ does not depend on the target y , the GGN is equal to the FIM (up to a constant scaling factor):

$$G(\theta_0) = N \tilde{F}(\theta_0). \quad (8.42)$$

In fact, the difference in the scaling factor has a natural interpretation. Previously, we saw that $F(\theta)$ is the FIM of the random variable (X, Y) distributed according to $p(x, y | \theta)$ and that the FIM is additive for independent random variables. Therefore, under this assumption, the GGN is equal to the FIM w.r.t. N i.i.d. observations $\{(x_n, y_n)\}_{n=1}^N$ distributed as $p(x, y | \theta)$ (and *not* as $p(x, y)$).

The requirement of the loss Hessian not depending on the target might seem arbitrary. However, this property is satisfied by a wide family of likelihoods, called **exponential families**.

Definition 8.1. A probability distribution with probability density function (PDF) of the form

$$p(y | w) = h(y) \exp(\phi(y)^{\top} w - \log Z(w)) \quad (8.43)$$

is called an **exponential family** of probability measures. The function $\phi: \mathcal{Y} \rightarrow \mathbb{R}^d$ is the vector of **sufficient statistics**. The parameters $w \in \mathbb{R}^d$ are the **natural parameters** of $p(y | w)$. The normalisation constant $Z(w): \mathbb{R}^d \rightarrow \mathbb{R}$ is the **partition function**. The function $h(y): \mathcal{Y} \rightarrow \mathbb{R}_{>0}$ is the **base measure**.

First, we state some facts about simple parameterisations.

Proposition 8.2. *For a probability distribution $p(y | w)$, $-\nabla_w \nabla_w^\top \log p(y | w)$ is independent of y if and only if $p(y | w)$ is an exponential family with natural parameter w .*

Parameterisations via neural networks are more delicate. The following proposition gives a necessary and sufficient condition for Equation 8.42.

Proposition 8.3. *For an exponential family $p(y | x, \theta)$ with natural parameters w given as a function of x and θ , $w(x, \theta) = (g \circ f)(x, \theta)$ for some function $g: \mathbb{R}^C \rightarrow \mathbb{R}^d$, the Hessian of $-\log p(y | x, \theta)$ w.r.t. $f(x, \theta)$ does not depend on y if and only if $\nabla_z \nabla_z^\top g(z)|_{z=f(x, \theta)} \cdot \phi(y)$ is constant in y where $\nabla_z \nabla_z^\top g(z)|_{z=f(x, \theta)} \in \mathbb{R}^{C \times C \times d}$ is a 3-tensor, and the product $\cdot$ contracts its third (d -dimensional) index against $\phi(y)$, giving a $C \times C$ matrix. A special case of the statement is when g is affine.*

The above statement on neural network parameterisations covers the most popular loss functions in classification and regression. In classification, the cross-entropy loss is the negative log-likelihood of a categorical distribution parameterised by our neural network. The logits are the categorical’s natural parameters, so with the identity link $g = \text{Id}$ the cross-entropy loss satisfies Proposition 8.3 in its affine special case. Similarly, the L_2 loss for regression satisfies Proposition 8.3 with $g = \text{Id}$.

A word of caution is in order: the GGN/FIM alone is usually degenerate; thus, it does not serve as a valid precision matrix for the Laplace approximation. Considering a single input-label pair (x_n, y_n) , the term

$$(J_\theta f(x_n, \theta_0))^\top \left(\nabla_z \nabla_z^\top \ell(z, y_n) \Big|_{z=f^{\theta_0}(x_n, \theta_0)} \right) J_\theta f(x_n, \theta_0) \quad (8.44)$$

is a positive *semidefinite* (PSD) matrix, as the loss Hessian $\nabla_z \nabla_z^\top \ell(z, y_n) \Big|_{z=f^{\theta_0}(x_n, \theta_0)}$ is positive semidefinite. For softmax cross-entropy, this Hessian is $\text{diag}(p) - pp^\top$, which is rank deficient as the softmax is shift-invariant. Usually, the GGN, that is, the sum of these matrices over the training dataset, is still rank deficient. However, adding the Hessian of the strongly convex regulariser term $\Omega(\theta_0)$ to the GGN results in the *positive definite* matrix

$$\Psi(\theta_0) = \nabla_\theta \nabla_\theta^\top \mathcal{L}_\Omega^{\theta_0}(\theta_0) = \nabla_\theta \nabla_\theta^\top \mathcal{L}^{\theta_0}(\theta_0) + \nabla_\theta \nabla_\theta^\top \Omega(\theta_0), \quad (8.45)$$

which is the Hessian of the regularised empirical risk of the linearised model, $\mathcal{L}_\Omega^{\theta_0}(\theta_0)$. This justifies using this matrix as a parameter-space precision in the Laplace approximation (Equation 7.4c).

With the curvature Ψ in hand, we can place the NTK of Section 8.2 and the posterior GP kernel of Equation 8.16 in a single frame. Both are Jacobian Gram matrices $J_\theta f M^{-1} J_\theta f^\top$ in a parameter-space metric M ; they differ in the expansion point (θ_0 for the NTK, θ_* for the posterior kernel) and in the metric M . Preconditioning the gradient flow of Equation 8.20 by M multiplies the parameter velocity by M^{-1} , inserting M^{-1} between the Jacobians and replacing its kernel k_t with $J_\theta f(x, \theta_t) M^{-1} J_\theta f(x_n, \theta_t)^\top$, holding the expansion point fixed and changing only the metric. Vanilla gradient descent uses the Euclidean metric $M = I$ and recovers the NTK, the isotropic-prior pushforward. Regularised natural-gradient descent [26] uses $M = \Psi(\theta_t)$, the GGN/Fisher curvature plus the prior precision (Equation 8.45) at the current parameters. We call the resulting kernel $J_\theta f(\cdot, \theta_t) \Psi(\theta_t)^{-1} J_\theta f(\cdot, \theta_t)^\top$ the **natural-gradient NTK**. Because $\Psi(\theta_t)$ is positive definite at every θ_t , the natural-gradient NTK governs the dynamics all along training, even where the gradient is nonzero and no Laplace posterior exists. It coincides with the posterior GP kernel $J_\theta f \Psi^{-1} J_\theta f^\top$ [813] only at a posterior mode θ_* , as the Laplace approximation requires. The parameter-space metric thus determines both the predictive uncertainty, through the Gaussian we push forward, and the optimisation geometry, through the descent the function follows.

Takeaways

For common losses (exponential-family with an *affine* link; e.g., squared loss, cross-entropy), the curvature of the linearised model satisfies

$$G(\theta_0) = N \tilde{F}(\theta_0),$$

identifying the GGN with the Fisher Information Matrix (FIM) over N i.i.d. samples, and tying the Riemannian and information-geometric views.

The Fisher Information Matrix is the Riemannian metric describing the local behaviour of the KL divergence:

$$D_{\text{KL}}(p(x, y | \theta_1) \parallel p(x, y | \theta_2)) = \frac{1}{2} \Delta \theta^\top F(\theta_1) \Delta \theta + \mathcal{O}(\|\Delta \theta\|^3).$$

Approx.	Storage	Cov.-vec.	Inversion	Correlations	Blind spot
Diag.	$\mathcal{O}(P)$	Elementwise division	Elementwise	None across parameters	Misses coupled directions
Low-rank	$\mathcal{O}(kP)$	Products with $Q_{1:k}$	Woodbury	Top eigenspace of G	Misses broad low-curvature directions
KFAC	$\sum_i \mathcal{O}(d_{i-1}^2 + d_i^2)$	Kronecker identities	Factor eigensystems	Within-layer Kronecker	Drops cross-layer couplings

Table 8.1: Common scalable curvature approximations for linearised Laplace. Here d_{i-1} and d_i are the input and pre-activation dimensions of layer i .

In practice, one uses the empirical input distribution to form the FIM. The matrix G is PSD and typically singular; adding a strongly convex regulariser makes

$$\Psi(\theta_0) = G(\theta_0) + \nabla_{\theta} \nabla_{\theta}^{\top} \Omega(\theta_0) \succ 0,$$

which is the invertible precision used by the Laplace approximation.

The metric M in the form $J_{\theta} f M^{-1} J_{\theta} f^{\top}$ sets which kernel we obtain: $M = I$ gives the NTK (vanilla gradient descent) and $M = \Psi$ gives the natural-gradient NTK (natural-gradient descent), which becomes the posterior GP kernel at a mode.

8.4 Scalable Curvature Approximations

The GGN of the linearised network (cf. Equation 8.22)

$$G(\theta) = \sum_{n=1}^N (J_{\theta} f(x_n, \theta))^{\top} \underbrace{\nabla_z \nabla_z^{\top} \ell(z, y_n) \big|_{z=f(x_n, \theta)}}_{=: H_n \succeq 0} J_{\theta} f(x_n, \theta) \in \mathbb{R}^{P \times P} \quad (8.46)$$

is dense in general and *far* too large to form for modern neural networks. We therefore seek structured surrogates that (i) are cheap to store and invert, yet (ii) retain meaningful parameter correlations. Table 8.1 compares the three curvature proxies we develop: the diagonal, low-rank, and Kronecker-Factored Approximate Curvature (KFAC) approximations. Scalable variants of these curvature approximations are developed further in Chapter 40, and last-layer inference is treated in Chapter 39.

A simple baseline is the diagonal $\text{diag } G(\theta)$, which is trivial to store and invert, though calculating it efficiently requires care. However, it encodes an independence assumption between *all parameters*, destroying the often-crucial within-layer correlations carried by $G(\theta)$. This tends to hurt the utility of the downstream uncertainty estimates [380].

8.4.1 Low-rank GGN Approximations

A scalable yet expressive alternative is a *low-rank* approximation of $G(\theta)$. As $G(\theta)$ is symmetric for any $\theta \in \mathbb{R}^P$, it possesses an eigendecomposition with real eigenvalues of the form

$$G(\theta) = Q(\theta) \Lambda(\theta) Q(\theta)^{\top}, \quad (8.47)$$

where $Q(\theta) \in \mathbb{R}^{P \times P}$ is an orthogonal matrix (i.e., a matrix with *orthonormal* columns/rows) and $\Lambda(\theta) \in \mathbb{R}^{P \times P}$ is a diagonal matrix.

For $q_{:,i}$, the i th column of $Q(\theta)$, and λ_i , the i th diagonal entry of $\Lambda(\theta)$, it holds that $G(\theta)q_{:,i} = \lambda_i q_{:,i}$. We follow the convention $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_P$. In the following, we drop the dependence of Q and its factors on θ for brevity.

We now seek a *good* low-rank approximation $\tilde{G}_k$ to G , as this reduces the sizes of the objects we need to handle. Since $G \succeq 0$, its eigenvalues are nonnegative, so the k algebraically largest coincide with the k largest in magnitude. The Eckart–Young–Mirsky theorem [484] states that the objective

$$\arg \min_{G' \in \mathbb{R}^{P \times P}, \text{rank}(G') \leq k} \|G - G'\| \quad (8.48)$$

is minimised by $\tilde{G}_k := Q_{1:k} \Lambda_{1:k} Q_{1:k}^\top$ where $Q_{1:k} \in \mathbb{R}^{P \times k}$ is the first k columns of Q and $\Lambda_{1:k} \in \mathbb{R}^{k \times k}$ holds the first k eigenvalues of G . This result holds for both the spectral norm $\|\cdot\|_2$ and the Frobenius norm $\|\cdot\|_F$. Therefore, one obtains an *optimal* low-rank approximation $\tilde{G}_k$ to G in the above sense by computing the first k eigenvalues and eigenvectors of G . Storing this low-rank approximation requires storing $(P+1)k$ values instead of P^2 .

The most naive way to compute $\tilde{G}_k$ is first calculating the eigendecomposition $G = Q \Lambda Q^\top$ and then truncating it to the first k eigenpairs. However, this strategy does not resolve the infeasibility of storing the full eigendecomposition with $P(P+1)$ values. Instead, one uses efficient *matrix-free* schemes such as the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method [907] or Lanczos iterations [969] to obtain the low-rank decomposition. Notably, these algorithms only require *matrix-vector products* $v \mapsto Gv$ with the GGN to obtain the top eigenpairs, which does not require representing the entire G matrix in memory.

The low-rank representation is particularly handy, as for simple priors, such as isotropic Gaussians, calculating the approximate covariance matrix is just as efficient as working with the low-rank GGN. Let $\Gamma \succ 0$ be the Hessian of the negative log-prior Ω , that is, the prior-curvature term $\nabla_\theta \nabla_\theta^\top \Omega$ of the precision $\Psi = G + \nabla_\theta \nabla_\theta^\top \Omega$ (Equation 8.45). Then the Woodbury matrix identity yields

$$(\Gamma + \tilde{G}_k)^{-1} = \Gamma^{-1} - \Gamma^{-1} Q_{1:k} (\Lambda_{1:k}^{-1} + Q_{1:k}^\top \Gamma^{-1} Q_{1:k})^{-1} Q_{1:k}^\top \Gamma^{-1}. \quad (8.49)$$

In particular, for isotropic Gaussians, $\Gamma = \tau I$,

$$(\tau I + Q_{1:k} \Lambda_{1:k} Q_{1:k}^\top)^{-1} = \tau^{-1} I + Q_{1:k} \text{diag} \left(-\frac{\lambda_1}{\tau(\lambda_1 + \tau)}, \dots, -\frac{\lambda_k}{\tau(\lambda_k + \tau)} \right) Q_{1:k}^\top. \quad (8.50)$$

In words: a diagonal-plus-low-rank matrix has a diagonal-plus-low-rank inverse. Multiplying by such a matrix is efficient and does not require forming a $P \times P$ matrix. For a vector $v \in \mathbb{R}^P$, we first calculate $u = Q_{1:k}^\top v \in \mathbb{R}^k$, then elementwise multiply u by the diagonal entries in $\text{diag} \left(-\frac{\lambda_1}{\tau(\lambda_1 + \tau)}, \dots, -\frac{\lambda_k}{\tau(\lambda_k + \tau)} \right)$, then map the result back to $\mathbb{R}^P$ with $Q_{1:k}$ and add the diagonal term $\tau^{-1}v$.

8.4.2 Kronecker-Factored Approximate Curvature

A complementary path towards scalable approximate curvature representations assumes independence *between layers* only, while retaining rich *within-layer* parameter correlations via a Kronecker factorisation. This idea underlies KFAC, which has recently seen wide use as a scalable curvature proxy [1136]. In this section, we give an introduction to KFAC for dense linear layers and refer to Dangel et al. [367] for a more detailed and general exposition of KFAC.

Consider layer i with weights $W^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$, input activation $a_n^{(i-1)} \in \mathbb{R}^{d_{i-1}}$, and pre-activation $s_n^{(i)} = W^{(i)} a_n^{(i-1)} \in \mathbb{R}^{d_i}$.⁴ Let $z_n = f(x_n, \theta) \in \mathbb{R}^C$ denote the network output. Let $J_{W^{(i)} s_n^{(i)}} \in \mathbb{R}^{d_i \times (d_i d_{i-1})}$ denote the Jacobian of $s_n^{(i)}$ w.r.t. $\text{vec}(W^{(i)})$, where vec flattens its input by stacking its columns. For datum n , define the *output-space* curvature $H_n := \nabla_z \nabla_z^\top \ell(z, y_n) \big|_{z=z_n} \succeq 0$, so $H_n \in \mathbb{R}^{C \times C}$. Writing $H_n = S_n S_n^\top$ with $S_n \in \mathbb{R}^{C \times C}$, the corresponding layer block of the GGN can be expressed as

$$\begin{aligned} G^{(i)} &= \sum_{n=1}^N (J_{W^{(i)} s_n^{(i)}})^\top \underbrace{(J_{s_n^{(i)} z_n}^\top H_n J_{s_n^{(i)} z_n})}_{=: \hat{H}_n^{(i)} \succeq 0} J_{W^{(i)} s_n^{(i)}} \\ &= \sum_{n=1}^N \sum_{c=1}^C (J_{W^{(i)} s_n^{(i)}})^\top g_{n,c}^{(i)} (g_{n,c}^{(i)})^\top J_{W^{(i)} s_n^{(i)}}. \end{aligned} \quad (8.51)$$

with $g_{n,c}^{(i)} := (J_{s_n^{(i)} z_n}^\top [S_n]_{:,c}) \in \mathbb{R}^{d_i}$, where $J_{s_n^{(i)} z_n}$ is the Jacobian of the network output z_n w.r.t. this layer's pre-activation $s_n^{(i)}$.⁵ Thus, the sum over c is over the network output curvature factor, while $g_{n,c}^{(i)}$ lives in the layer pre-activation space.

For a linear layer, the output-parameter Jacobian has the form $J_{W^{(i)} s_n^{(i)}} = (a_n^{(i-1)})^\top \otimes I_{d_i}$ where $\otimes$ denotes the Kronecker product. Noting the Kronecker properties $(A \otimes B)^\top = A^\top \otimes B^\top$ and $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

⁴Bias terms are omitted for brevity.

⁵For the squared loss, $H_n = I$; for cross-entropy, $H_n = \text{diag}(\text{softmax}(z)) - \text{softmax}(z) \text{softmax}(z)^\top$ is the softmax Hessian.

for compatible matrices A, B, C, D , we get

$$G^{(i)} = \sum_{n=1}^N \sum_{c=1}^C \underbrace{(a_n^{(i-1)}(a_n^{(i-1)})^\top)}_{\text{input factor}} \otimes \underbrace{(g_{n,c}^{(i)}(g_{n,c}^{(i)})^\top)}_{\text{backpropagated factor}} \in \mathbb{R}^{(d_{i-1}d_i) \times (d_{i-1}d_i)}. \quad (8.52)$$

The KFAC approximation then pulls the Kronecker product outside the sum:

$$\begin{aligned} & \sum_{n=1}^N \sum_{c=1}^C \left(a_n^{(i-1)}(a_n^{(i-1)})^\top \right) \otimes \left(g_{n,c}^{(i)}(g_{n,c}^{(i)})^\top \right) \\ & \approx \underbrace{\left(\sum_{n=1}^N a_n^{(i-1)}(a_n^{(i-1)})^\top \right)}_{=: A^{(i)}} \otimes \underbrace{\left(\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C g_{n,c}^{(i)}(g_{n,c}^{(i)})^\top \right)}_{=: B^{(i)}}. \end{aligned} \quad (8.53)$$

Thus, the layer block is approximated by

$$G^{(i)} \approx A^{(i)} \otimes B^{(i)} =: \tilde{G}^{(i)}, \quad A^{(i)} \in \mathbb{R}^{d_{i-1} \times d_{i-1}}, B^{(i)} \in \mathbb{R}^{d_i \times d_i}. \quad (8.54)$$

Stacking layers and *neglecting inter-layer couplings* yields a block-diagonal curvature with Kronecker blocks $A^{(i)} \otimes B^{(i)}$.

The Kronecker form is efficient to store. While the dense GGN $G^{(i)}$ for a single linear layer has $(d_{i-1}d_i)^2$ entries, the Kronecker factorisation only requires storing $A^{(i)}$ and $B^{(i)}$, that is, a total of $d_{i-1}^2 + d_i^2$ values. The block-diagonal layer approximation then avoids storing cross-layer blocks. One can also calculate efficient covariance-vector products without forming a dense $P \times P$ matrix by leveraging the cached eigendecompositions of the Kronecker factors and standard Kronecker identities. This property is more involved to show than in the case of low-rank approximations; refer to Dangel et al. [367] for more details.

Takeaways

KFAC assumes independence *across* layers but preserves rich *within-layer* correlations through a Kronecker factorisation of each layer's curvature block: $G^{(i)} \approx A^{(i)} \otimes B^{(i)}$. It is far more expressive than diagonal curvature structures and complementary to low-rank approximations (usually at lower cost). Storing the Kronecker factors is cheap. With an isotropic prior precision, covariance-vector products can be implemented remarkably efficiently. See Dangel et al. [367] for implementation details and extensions.

8.5 The Predictive

Equation 8.13 gives a Gaussian distribution in the output space for input x_* . For regression, this distribution is called the **predictive distribution**: its mean encodes the average prediction under the parameter-space posterior and the linearised model, and its covariance structure quantifies the uncertainty in the prediction, including correlations of the outputs. Figure 8.3 shows a one-dimensional regression example of the linearised Laplace approximation.

In classification, however, we are interested in obtaining a second-order distribution (a distribution over distributions) over the *categorical* distributions of a discrete set of labels. To this end, one has to map the logit-space Gaussian measure in Equation 8.13 through an activation function onto the $(C-1)$ -dimensional probability simplex:

$$h: \mathbb{R}^D \xrightarrow{f} \mathcal{G}(\mathbb{R}^C) \xrightarrow{\varphi_*} \mathcal{P}(\Delta^{C-1}) \quad (8.55)$$

where f is the mapping from an input x_* to the logit-space Gaussian distribution given by the Laplace-approximated network, $\mathcal{G}(\mathbb{R}^C)$ is the set of Gaussian measures on the logit space $\mathbb{R}^C$, $\mathcal{P}(\Delta^{C-1})$ is the set of probability measures on the simplex, and φ_* pushes forward the Gaussian probability measures through the activation function. The roman f is the network with its parameter posterior pushed through it: the affine pushforward of the Laplace posterior $\mathcal{N}(\theta; \theta_*, \Psi(\theta_*)^{-1})$ along the linearised network, giving $f(x_*) = \mathcal{N}(f(x_*, \theta_*), J_\theta f(x_*, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x_*, \theta_*)^\top) \in \mathcal{G}(\mathbb{R}^C)$ as in Equation 8.13. The italic $f(x_*, \theta_*) \in \mathbb{R}^C$ plugs in the point estimate θ_* and returns a single logit vector; the roman $f(x_*)$ returns the full logit-space Gaussian, whose mean is exactly that italic vector. All parameters in the support of the Laplace posterior are taken into account rather than a single one being held fixed, so f takes

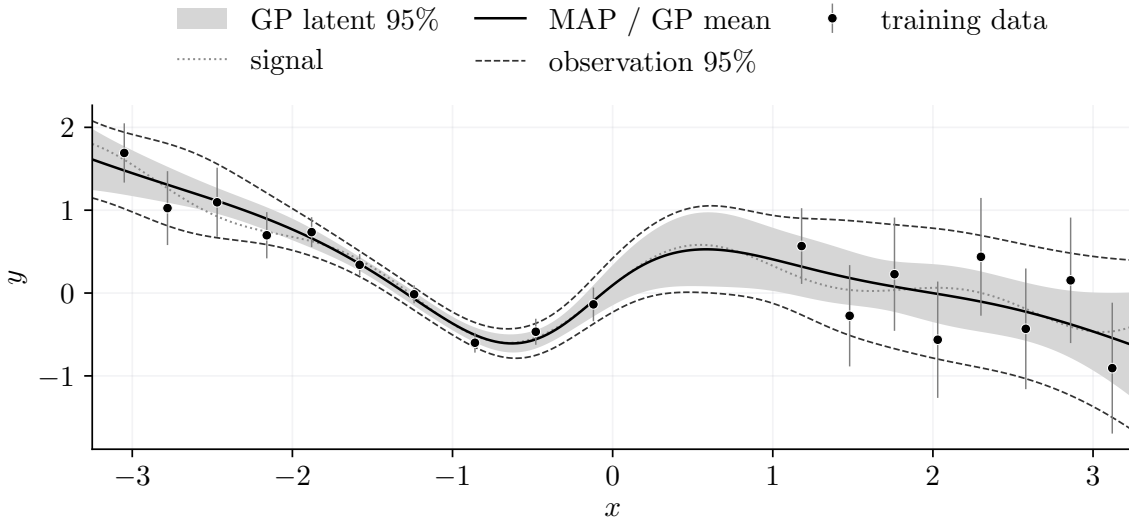


Figure 8.3: A one-dimensional regression example with irregular inputs and input-dependent observation noise. The model is a neural network with a Bayesian last layer, so the linearisation is exact and the linearised GP yields the exact posterior predictive. The shaded region shows the pointwise 95% latent GP interval, while the dashed curves add the input-dependent observation variance.

only the input. In words, h is the composition of the Laplace-approximated neural network outputting logit-space Gaussian distributions, f , and the pushforward map through the activation function that gives us a distribution over probability vectors.

In most cases, this activation function is the softmax, defined as $\text{softmax}: \mathbb{R}^C \rightarrow \Delta^{C-1}$,

$$\text{softmax}_c(z) = \frac{\exp(z_c)}{\sum_{c'=1}^C \exp(z_{c'})}. \quad (8.56)$$

In this case, there exists no closed-form expression for $h(x_*) \in \mathcal{P}(\Delta^{C-1})$ given input x_* . Further, $\mathbb{E}_{P \sim h(x_*)}[P]$, called the posterior predictive, is also intractable. However, sampling from $h(x_*)$ is efficient, as it only requires sampling from the logit-space Gaussian $f(x_*)$ and pushing these logit samples forward through the activation function. Thus, in practice, one resorts to Monte Carlo integration, that is, drawing M samples from $h(x_*)$ and calculating their empirical average. Similarly, one can approximate other derived quantities, such as the expected entropy $\mathbb{E}_{P \sim h(x_*)}[\mathcal{H}(P)]$, through Monte Carlo sampling. Such derived quantities are often used in the field of uncertainty disentanglement, where different quantities capture different sources of uncertainty [1204]. These uncertainty estimates are applied to selective classification in Chapter 44.

The main issue with the above schemes is the requirement of sampling: the error of Monte Carlo integration decreases as the inverse square root of the number of samples M , meaning that to reduce the error by three orders of magnitude, one needs a million samples. Standard softmax-cross-entropy integration cannot avoid this sampling cost without incurring significant approximation error. However, some output activation functions admit a *closed-form* posterior predictive. For a single output, integrating a normCDF (probit) link against a Gaussian is exact, and the logistic sigmoid admits a classical probit approximation [1106]. Mucsányi et al. [1205] show that the normalised elementwise sigmoid and normCDF functions yield closed-form predictives that have strong approximation guarantees and are naturally compatible with more performant training objectives than the standard softmax-cross-entropy objective. Further, derived quantities such as the expected entropy can also be obtained in closed form by matching the moments of the Gaussian pushforward distributions with Dirichlets. The change in the training objective comes at no cost: models trained this way become better calibrated while providing sample-free posterior predictives and other derived quantities. Therefore, Laplace approximations in classification can be formulated in a purely sample-free fashion, with theoretical guarantees. Figure 8.4 compares the MAP, Monte Carlo softmax, and sample-free probit predictives on a two-dimensional example, together with the predictive entropy.

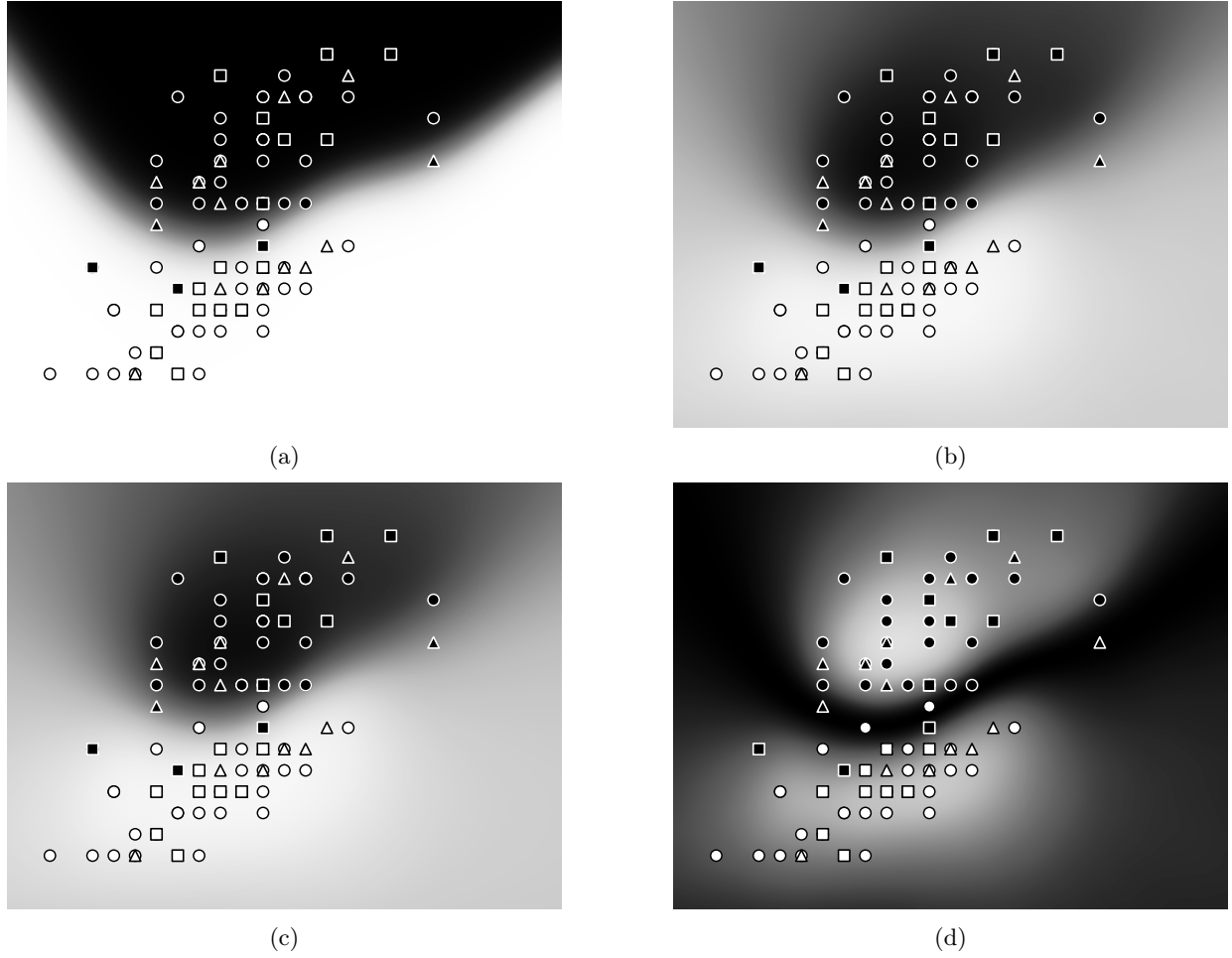


Figure 8.4: Two-dimensional binary classification with a Laplace approximation to the last layer of a fixed-feature network on the Iris dataset. The inputs are the measured sepal width and petal width. White denotes Iris versicolor and black denotes Iris virginica. Circles are training points, triangles are validation points, and squares are test points. The prior precision is selected on the validation set by log score. Panels (a)–(c) show the probability of the black class for the MAP predictor, the softmax posterior predictive estimated by Monte Carlo sampling, and the sample-free probit approximation. Panel (d) shows the predictive entropy computed from the Monte Carlo predictive. On the held-out test points, the MAP log score and Brier score are 0.433 and 0.122, while the Monte Carlo GP predictive gives 0.345 and 0.107, and the probit predictive gives 0.347 and 0.108—close to indistinguishable.

Takeaways

For regression, the linearised Laplace output at input x_* is *Gaussian* (Equation 8.13); its mean is $f(x_*, \theta_*)$ with covariance $J_{\theta} f(x_*, \theta_*) \Psi(\theta_*)^{-1} J_{\theta} f(x_*, \theta_*)^{\top}$.

In classification with softmax, pushing this logit-space Gaussian through the activation yields a distribution on the simplex with *no closed form* for the posterior predictive $\mathbb{E}_{P \sim h(x_*)}[P]$, so one typically uses Monte Carlo with $\mathcal{O}(M^{-1/2})$ error.

Replacing softmax by a normalised elementwise sigmoid or normCDF output activation makes the posterior predictive (and derived quantities such as expected entropy via Dirichlet moment matching) *closed-form* with approximation guarantees, usually improving calibration and rendering Laplace classification *sample-free*.

Chapter 9

The differential geometric perspective on Laplace

Linearised Laplace approximations (Section 8.1), and specifically their covariance structure, have interpretations in Riemannian and information geometry. They encode the neural network’s parameter space geometry, as informed by the loss.

9.1 The GGN Metric

In the previous chapter, we discussed in detail the curvature matrix defined in Equation 8.4, called the Generalised Gauss-Newton matrix (GGN). It is the precision term from the likelihood and, combined with the precision term from the prior and inverting, it yields the covariance matrix of a linearised Laplace approximation.

Recall that the GGN matrix at the parameters $\theta \in \mathbb{R}^P$ is given by (Equation 8.22)

$$G(\theta) = \nabla_{\theta} \nabla_{\theta}^{\top} \mathcal{L}^{\theta}(\theta) = \sum_{n=1}^N (J_{\theta} f(x_n, \theta))^{\top} \left(\nabla_z \nabla_z^{\top} \ell(z, y_n) \Big|_{z=f(x_n, \theta)} \right) J_{\theta} f(x_n, \theta). \quad (9.1)$$

We will now see how the GGN arises as a metric on the parameter space. Recall our training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1, \dots, N}$. We restrict the neural network $f: \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$ to the training set, to obtain the function

$$\begin{aligned} f_X: \mathbb{R}^P &\rightarrow (\mathbb{R}^C)^N \\ \theta &\mapsto (f(x_n, \theta))_{n=1}^N. \end{aligned} \quad (9.2)$$

For a single input-label pair (x_n, y_n) , assuming $\ell(z, y_n)$ is strictly convex (which is the case, for example, for the squared loss in regression), the positive definite loss Hessian

$$H_n(z) := \nabla_z \nabla_z^{\top} \ell(z, y_n) \quad (9.3)$$

characterises a **Riemannian metric** in the output space $\mathbb{R}^C$. That is, it defines a symmetric positive definite bilinear form $(v_1, v_2) \mapsto v_1^{\top} H_n(z) v_2$ at each point $z \in \mathbb{R}^C$. Such a metric encodes the *curvature* of the loss given the target y_n . Note that if the likelihood is an exponential family, then H_n is independent of the target y_n (Proposition 8.2).

In fact, we only require $\ell(z, y_n)$ to be (not necessarily strictly) convex, in which case H_n is only positive semidefinite. This is the case for the cross-entropy loss in classification, where the Hessian is rank-deficient. In this case, H_n defines a *degenerate* Riemannian metric on $\mathbb{R}^C$.

We can then equip the image of f_X , $(\mathbb{R}^C)^N$, with the block diagonal (degenerate) Riemannian metric

$$H(z) = \begin{bmatrix} H_1(z_1) & 0 & \cdots & 0 \\ 0 & H_2(z_2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & H_N(z_N) \end{bmatrix} \quad (9.4)$$

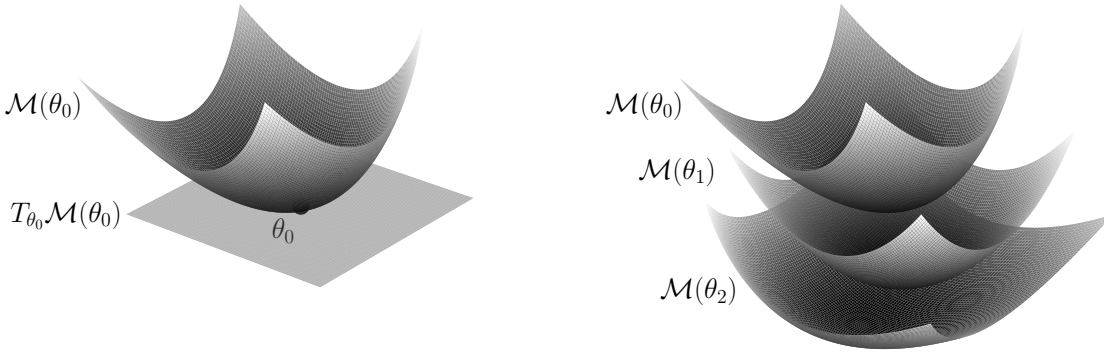


Figure 9.1: *Left:* A fiber manifold $\mathcal{M}(\theta_0)$ in the parameter space, and its tangent space $T_{\theta_0}\mathcal{M}(\theta_0)$ at θ_0 . The fiber manifold consists of the parameters yielding the same output values on the training data as θ_0 (Equation 9.7), the tangent space consists of the kernel directions of the GGN at θ_0 . *Right:* A collection of three fibers in parameter space. The parameter space is partitioned into a disjoint union of such fibers.

in standard coordinates, where $z = (z_1, \dots, z_N)$. This metric characterises the curvature of the loss on the training set. It can then be pulled back through f_X to the parameter space:

$$\begin{aligned}
 (f_X^* H)(\theta) &= J_\theta f_X(\theta)^\top H(f_X(\theta)) J_\theta f_X(\theta) \\
 &= \sum_{n=1}^N J_\theta f(x_n, \theta)^\top H_n(f(x_n, \theta)) J_\theta f(x_n, \theta) \\
 &= \sum_{n=1}^N (J_\theta f(x_n, \theta))^\top \left(\nabla_z \nabla_z^\top \ell(z, y_n) \Big|_{z=f(x_n, \theta)} \right) J_\theta f(x_n, \theta) = G(\theta),
 \end{aligned} \tag{9.5}$$

i.e., the **pullback** $f_X^* H$ corresponds exactly to the GGN G .

Due to symmetries in the neural network architecture (see Part VII) and overparametrisation, the GGN is rank-deficient and only positive *semidefinite*. It therefore defines again a *degenerate* Riemannian metric on the parameter space.¹ Thus, *damping* is needed to get a proper Riemannian metric on the parameter space. The strictly convex regulariser term $\Omega(\theta)$ is such a dampener: adding its Hessian to the GGN results in a positive definite matrix (cf. Equation 8.45)

$$\nabla_\theta \nabla_\theta^\top \mathcal{L}_\Omega^\theta(\theta) = \nabla_\theta \nabla_\theta^\top \mathcal{L}^\theta(\theta) + \nabla_\theta \nabla_\theta^\top \Omega(\theta). \tag{9.6}$$

Using this as a preconditioner in second-order optimisation means we perform Riemannian (stochastic) gradient descent in the parameter space.

But how do we interpret the GGN metric? We base our presentation on Roy et al. [1427]. Fixing a parameter vector θ_0 , consider the set

$$\mathcal{M}(\theta_0) = \{\theta \in \mathbb{R}^P : f_X(\theta) = f_X(\theta_0)\} \tag{9.7}$$

i.e., the set of parameters θ for which the neural network defines the same function on the training data as θ_0 . This is an equivalence class of parameters, called a **fiber**. By the preimage theorem [1640, Theorem 9.9], it is (locally) a manifold of dimension $P - NC$ if f_X is (locally) continuously differentiable and the Jacobians $J_\theta f_X(\theta)$ are surjective for all (local) $\theta \in \mathcal{M}(\theta_0)$. Intuitively, the parameter space is P -dimensional but we constrain the model's prediction on the training set of N samples, meaning we put NC scalar constraints on the output. This eliminates NC degrees of freedom, yielding a $(P - NC)$ -dimensional manifold. The surjectivity condition holds in particular when the eigenvalues of the neural tangent kernel matrix over the entire training set are bounded away from 0, a common assumption for overparametrised models, see Roy et al. [1427, Appendix A] and the references therein.

Further, when the Jacobians $J_\theta f_X(\theta)$ are surjective locally around θ_0 , the preimage theorem further tells us that $\ker J_\theta f_X(\theta_0) = T_{\theta_0}\mathcal{M}(\theta_0)$, the tangent space of $\mathcal{M}(\theta_0)$ at θ_0 (see Figure 9.1). If the Hessians $\nabla_z \nabla_z^\top \ell(z, y_n)$ are full rank (for example, when $\ell(z, y_n) = \|z - y_n\|^2$ the squared loss), we get $\ker G(\theta_0) = T_{\theta_0}\mathcal{M}(\theta_0)$. These

¹Not to be confused with pseudo-Riemannian metrics.

are directions in which the training data gives us no information about the parameters. Thus, in such directions, linearised Laplace approximations solely rely on the prior. The orthogonal complement of this null space consists of the directions in which the training data gives us information.

Now having defined the fibers in parameter space, we can quotient them out to obtain the **effective parameter space**: $\mathbb{R}^P / \sim$ where $\sim$ is the equivalence relation induced by f_X , $\theta_0 \sim \theta_1 \Leftrightarrow f_X(\theta_0) = f_X(\theta_1)$. The effective parameters are parameters for which the neural network f takes *distinct* values on the training data. Formally, the effective parameter space is a *fiber bundle* over the parameter space. It can be thought of as a manifold perpendicular to the fibers $\mathcal{M}(\theta)$; Roy et al. [1427] construct such a manifold explicitly. We note, however, that this is an arbitrary choice, as there is no canonical way to choose representatives for the equivalence classes and hence such a manifold. As a consequence, the non-linear effective parameter space is difficult to leverage computationally. However, the GGN's degenerate metric on the original parameter space $\mathbb{R}^P$ captures the similarity of training outputs between different parameters. One therefore does not have to construct the effective parameter space explicitly, but rather use the GGN's geometry in the parameter space.

Takeaways

The GGN is the **pullback** through the restricted network f_X of the loss-curvature metric on the training outputs,

$$G(\theta) = (f_X^* H)(\theta) = J_\theta f_X(\theta)^\top H(f_X(\theta)) J_\theta f_X(\theta),$$

and therefore defines a (degenerate) Riemannian metric on the parameter space $\mathbb{R}^P$. Its null space coincides with the tangent space $T_{\theta_0} \mathcal{M}(\theta_0)$ of the *fiber* $\mathcal{M}(\theta_0)$ of parameters producing identical outputs on the training data: these are the directions in which the data is uninformative and a linearised Laplace approximation falls back on the prior. Damping (e.g. a strictly convex regulariser Ω) turns this degenerate metric into a proper one.

9.2 The Fisher Metric

In the previous chapter we saw that the GGN metric can be seen as the pullback of the loss curvature metric on the training outputs. We now ask whether there is a more intrinsic, parametrisation-independent justification for this metric. We offer a such a perspective on the GGN metric via information geometry.

Recall from Chapter 8, Equation 8.26 the **Fisher information metric**

$$\begin{aligned} F(\theta) &= \mathbb{E}_{p(x,y|\theta)} [-\nabla_\theta \nabla_\theta^\top \log p(x, y | \theta)] \\ &= \mathbb{E}_{p(x,y|\theta)} [\nabla_\theta \log p(x, y | \theta) (\nabla_\theta \log p(x, y | \theta))^\top] \end{aligned} \quad (9.8)$$

which we now view as a degenerate Riemannian metric on the parameter space $\mathbb{R}^P$. If we take the empirical distribution on the x , $\frac{1}{N} \sum_{n=1}^N \delta_{x_n}$, we get (Equations 8.37 and 8.40)

$$\begin{aligned} \tilde{F}(\theta) &= \frac{1}{N} \sum_{n=1}^N \mathbb{E}_{p(y|x,\theta)} [-\nabla_\theta \nabla_\theta^\top \log p(y | x, \theta)] \\ &= \frac{1}{N} \sum_{n=1}^N \nabla_\theta \log p(y_n | x_n, \theta) (\nabla_\theta \log p(y_n | x_n, \theta))^\top. \end{aligned} \quad (9.9)$$

We saw in Chapter 8 that $\tilde{F}(\theta) = \frac{1}{N} G(\theta)$, a renormalisation of the GGN metric, under the mild condition that the Hessian of the negative log-likelihood loss does not depend on the target y (see Equation 8.42).

This gives an important connection between the covariance used in linearised approximation and the fundamental object from information geometry, which is the Fisher information metric. The goal of this chapter is to explain the importance of the Fisher information metric, why it can be seen as the natural metric for any probabilistic model, and its interpretation in deep learning.

A **statistical manifold** is a manifold $\mathcal{M}$ consisting of probability measures. Any parametric statistical model defines a parametrisation of a statistical manifold $\mathbb{R}^P \rightarrow \mathcal{M}$. What is $\mathcal{M}$ for a neural network? So far, we have viewed a neural network as a map $f: \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathbb{R}^C$. However, recall that we assume it induces probability distributions $p(y | x, \theta)$, since the loss function $\ell(f(x, \theta), y)$ is assumed to be the negative log-likelihood $-\log p(y | x, \theta)$. Therefore we implicitly viewed f as a probabilistic model; for each $x \in \mathbb{R}^D$, and $\theta \in \mathbb{R}^P$, we have a probability

measure $f(x, \theta) \in \mathcal{P}(\mathcal{Y})$ defined by the neural network, where $\mathcal{Y}$ is the set of labels ($\mathbb{R}^C$ for regression, $\{1, \dots, C\}$ for classification) and $\mathcal{P}(\mathcal{Y})$ is the space of probability measures on the set of labels. f is given by the composition of f with some appropriate $g: \mathbb{R}^C \rightarrow \mathcal{P}(\mathcal{Y})$. In regression, we most often take $g(z) = \mathcal{N}(z, \epsilon^2 I) \in \mathcal{P}(\mathbb{R}^C)$ for some fixed $\epsilon > 0$, while in classification, most often $g(z) = \text{softmax}(z) \in \Delta^{C-1} \cong \mathcal{P}(\{1, \dots, C\})$. In Chapter 8, we saw that both of these choices guarantee the equivalence between the Fisher and the GGN (Equation 8.42).

So we have a map $f: \mathbb{R}^D \times \mathbb{R}^P \rightarrow \mathcal{P}(\mathcal{Y})$. Write $p(y | x, \theta)$ for the density of $f(x, \theta)$, so $p(y | x, \theta) = \frac{1}{\sqrt{2\pi\epsilon}} \exp(-\frac{1}{2\epsilon^2} \|f(x, \theta) - y\|^2)$ in regression, $p(y | x, \theta) = \text{softmax}(f(x, \theta))_y$ in classification. Then the **statistical model** $m: \mathbb{R}^P \rightarrow \mathcal{P}(\mathbb{R}^D \times \mathcal{Y})$ is defined as the joint distribution over x and y . Namely, if the distribution over the inputs x has a density $p(x)$, the density of $m(\theta)$ is $p(x, y | \theta) := p(x)p(y | x, \theta)$. As discussed above, in practice we take the empirical distribution over x . The statistical manifold $\mathcal{M}$ defined by the neural network f is the image of m in $\mathcal{P}(\mathbb{R}^D \times \mathcal{Y})$.

This is where the Fisher information metric comes into play. The Fisher metric is an object *intrinsic* to the statistical manifold $\mathcal{M}$, and is independent of how the manifold is parametrised by $\mathbb{R}^P$. Formally, suppose we have a different parametrisation $\tilde{m}: \mathbb{R}^P \rightarrow \mathcal{M}$. Then m and $\tilde{m}$ differ by some diffeomorphism² of the parameter space, i.e. there is a diffeomorphism $\varphi: \mathbb{R}^P \rightarrow \mathbb{R}^P$ such that $m = \tilde{m} \circ \varphi$. Then we see that for $\theta = \varphi(\tilde{\theta})$, $m(\theta) = \tilde{m}(\tilde{\theta})$ or, abusing notation, $p(x, y | \theta) = p(x, y | \tilde{\theta})$. Therefore, from the definition of the Fisher information metric (Equation 9.8 or Equation 9.9), the Fisher information metric on the two distinct parametrisations coincide $F_m(\theta) = F_{\tilde{m}}(\tilde{\theta})$.

The fact that the Fisher metric does not depend on the parametrisation implies that it is intrinsic to $\mathcal{M}$. Therefore, there should exist a way to define it directly on $\mathcal{M}$ in an abstract, measure-theoretic way. Doing this rigorously requires some mathematical groundwork, which the keen reader can find in Ay et al. [66]. Briefly, for $\mu \in \mathcal{M}$ and tangent vectors $\nu_1, \nu_2 \in T_\mu \mathcal{M}$, which are signed measures integrating to 0, the intrinsic Fisher information metric is defined by

$$F(\mu)[\nu_1, \nu_2] = \int \frac{d\nu_1}{d\mu} \frac{d\nu_2}{d\mu} d\mu \quad (9.10)$$

where $\frac{d\nu_i}{d\mu}$ is the Radon-Nikodym derivative of ν_i with respect to μ , i.e. the densities of ν_i with respect to μ . Then the degenerate metric on the parameter space $\mathbb{R}^P$ is given by the pullback of this intrinsic metric onto $\mathbb{R}^P$ (see Ay et al. [66]).

There is an infinite number of ways to define an intrinsic Riemannian metric on $\mathcal{M}$. So what makes the Fisher information metric special? It is not special for its invariance under transformations of the *parameter space*, but instead for its invariance under transformations of the *data space*.

Definition 9.1. Let $\mathcal{D}$ be a measurable space, thought of as a data space. A statistic is a measurable map $T: \mathcal{D} \rightarrow \mathcal{D}'$, where $\mathcal{D}'$ is some other measurable space.

Let $\mathcal{M}$ be a statistical manifold over $\mathcal{D}$, that is, consisting of probability measures over $\mathcal{D}$. A statistic $T: \mathcal{D} \rightarrow \mathcal{D}'$ is called *sufficient* if the measures in $\mathcal{M}$ have densities $q(T(x), \theta)h(x)$ with respect to some base measure, for $x \in \mathcal{D}$ and θ parametrising $\mathcal{M}$.

The factorisation condition means that the likelihood depends on θ only through $T(x)$, since the factor $h(x)$ has no θ -dependence. Thus, all information about θ contained in the full data point x is also contained in the statistic $T(x)$, justifying the term ‘sufficient’.

In our deep learning setting above, the data space is $\mathcal{D} = \mathbb{R}^D \times \mathcal{Y}$, the product of the input space with the label space.

Note that, for a statistical manifold $\mathcal{M}$ over $\mathcal{D}$, a statistic $T: \mathcal{D} \rightarrow \mathcal{D}'$ induces a map $T_*: \mathcal{M} \rightarrow \mathcal{M}'$ acting by pushforwards: $T_*\mu = \mu \circ T^{-1}$. So $\mathcal{M}'$ is a statistical manifold over $\mathcal{D}'$. Then, under mild assumptions on $\mathcal{M}$, we obtain the following theorem:

Theorem 9.1 (Proposition 5.6 in Ay et al. [66]). Suppose $\mathcal{M}$ has a positive regular density function [66, Definition 3.5]. Then the Fisher information metric is invariant under sufficient statistics $T: \mathcal{D} \rightarrow \mathcal{D}'$, i.e. $T_*: \mathcal{M} \rightarrow \mathcal{M}'$ is an isometry of Riemannian manifolds, when $\mathcal{M}$ and $\mathcal{M}'$ are equipped with their respective Fisher information metrics.

Up to a multiplicative constant, the Fisher information metric is the unique metric satisfying this. This is Chentsov’s theorem. We leave as an exercise to verify this theorem for the special case of the deep learning setting above.

As explained above, sufficient statistics should be thought of as maps of the data space which do not induce a loss of information on the parameters with respect to a given statistical model. The Fisher information metric,

²A diffeomorphism is a smooth bijection with smooth inverse.

being invariant under such transformations, should be thought of as quantifying ‘information’, hence its name. Note in particular that it is invariant under diffeomorphisms of the data space (which are trivially sufficient statistics owing to their bijective nature), and so does not depend on how the data is ‘presented’. This makes it the natural metric for statistical manifold $\mathcal{M}$.

Takeaways

Up to the rescaling $\tilde{F}(\theta) = \frac{1}{N}G(\theta)$, the GGN *is* the empirical Fisher information metric — the intrinsic metric on the statistical manifold $\mathcal{M}$ of distributions defined by the network. Being intrinsic, it is **independent of how $\mathcal{M}$ is parametrised**, and by *Chentsov’s theorem* it is the unique metric (up to scale) invariant under sufficient statistics, i.e. under transformations of the *data* space that lose no information about the parameters. This singles out the Fisher as the natural, ‘information-theoretic’ metric underlying the GGN.

9.3 Geometric Laplace Approximations

Laplace approximations, being Gaussian approximations to the neural network’s posterior, are limited in their expressivity. As a consequence, there have been attempts at leveraging the geometry of the parameter space to define more expressive variants of Laplace approximations.

A key observation is that the Laplace approximations are not diffeomorphism-invariant: given a diffeomorphism of the parameter space $\varphi: \mathbb{R}^P \rightarrow \mathbb{R}^P$, if such diffeomorphism is non-linear, then performing Laplace with respect to the parameters φ , i.e., a **reparametrised Laplace approximation**, yields a distinct posterior approximation than performing Laplace with respect to the parameters θ . In the extreme case, one can show there exists a diffeomorphism φ for which the Laplace approximation with respect to the corresponding parameters is exact [333]. In practice, computing such φ is as difficult as computing the exact posterior itself. However, this result gives hope that one may be able to construct computationally feasible diffeomorphisms φ yielding high-quality Laplace approximations, see for instance MacKay [1109]. An important note is that, while Laplace approximations are not invariant under diffeomorphisms of the parameter space, the predictives of linearised Laplace approximations are, up to changing the prior. We leave this as an exercise to the reader.

Another attempt at leveraging geometry for Laplace is to perform the Laplace approximation in the tangent space of the parameter space and then project it through an exponential map under some Riemannian metric [121, 1823], coined **Riemannian Laplace approximations**. The latter work uses the Fisher as the Riemannian metric. Work remains to be done to make the computation of the exponential map efficient and scalable. See Costa et al. [333] for the relationship between such Riemannian Laplace approximations and the aforementioned reparametrised Laplace approximations.

Finally, **diffusion Laplace approximations** [1427] sample the approximate posterior through random walks with respect to a Riemannian metric in the weight space. Once again, while there are promising preliminary results, further work is needed to scale such approaches computationally. Moreover, the relationship between diffusion Laplace approximations and Riemannian Laplace approximations is yet to be described.

Takeaways

Laplace approximations are **not invariant** under non-linear reparametrisations $\varphi: \mathbb{R}^P \rightarrow \mathbb{R}^P$ of the parameter space — in the extreme, some φ even makes Laplace exact. This freedom motivates *geometric* variants: **reparametrised**, **Riemannian** (Laplace in the tangent space pushed through an exponential map, often under the Fisher metric), and **diffusion** Laplace approximations. All exploit the parameter-space geometry for more expressive posteriors, but scalable computation — of suitable diffeomorphisms, exponential maps, and random walks — remains an open problem.

Chapter 10

Analytic, low-cost functionality from Laplace

In the previous sections, we have seen how the Laplace approximation endows a neural network with a Gaussian weight-space uncertainty (cf. Chapter 7), and how this uncertainty can be transformed into a closed-form expression for predictive uncertainty (cf. Chapter 8) informed by the training data (cf. Chapter 9). By combining the linearised pushforward with analytic approximations for the softmax integral in classification, we obtain a versatile analytic toolbox for uncertainty-driven decision making.

Classical Bayesian statistical methods – such as probabilistic prediction, out-of-distribution detection, active learning, Bayesian optimisation, continual learning, and model selection – become directly accessible in deep learning under the Gaussian assumption. This setting also enables further approximations for inference at various computational budgets [380, 1730].

In this chapter, we examine two case studies demonstrating how such uncertainty estimates can be leveraged to address open challenges in deep learning. The first focuses on how to mitigate predictive overconfidence in ReLU networks. The second addresses continual learning: we will see that the Laplace approximation provides a natural regularisation term, which alleviates catastrophic forgetting. We then conclude with additional observations on how the Laplace-Gaussian Process connection can be leveraged for optimal data selection under different objectives. For a discussion on computational techniques for scaling these approaches to large networks, such as KFAC and subnetwork approximations, we refer to Section 8.4 and Chapter 40.

Takeaways

Since linearised Laplace turns virtually every deep neural network into a Gaussian process predictive distribution, the powerful practical functionalities of Gaussian process regression – confidence and uncertainty quantification; model adaptation; active learning and design; continual learning; etc. – all become directly available in deep learning. Of course, since this is an approximate framework, these functionalities can only be expected to work to varying degrees of approximation.

10.1 Fixing overconfidence with calibrated uncertainty

10.1.1 Provable overconfidence in ReLU-like neural networks

In supervised classification, deep neural networks are trained to map an input $x \in \mathbb{R}^d$ to a vector of class logits $f_\theta(x) \in \mathbb{R}^C$, which are converted into categorical probabilities via the softmax function (8.56). Ideally, these probabilities should reflect the true uncertainty of the model’s predictions. In practice, however, modern deep networks tend to be *overconfident*: they assign very high probabilities to certain classes even when the input is far from the training distribution. This behaviour is particularly problematic in safety-critical settings, such as medical diagnosis or autonomous driving, where unwarranted certainty can have serious consequences [729].

A precise theoretical explanation for this behaviour in ReLU-like networks was given by Hein et al. [729]. Such networks consist of linear layers (fully connected or convolutional) combined with piecewise-linear activations such

as ReLU or LeakyReLU. Consequently, they implement a piecewise affine mapping:

$$f(x) = \mathbf{V}^{(l)}x + \mathbf{a}^{(l)}, \quad x \in \mathcal{Q}_l, \quad l = 1, \dots, R$$

where the polytopes $\mathcal{Q}_1, \dots, \mathcal{Q}_R$ partition the input space $\mathbb{R}^d$ into regions of constant activation pattern. Within each such region, the logits grow linearly with $\|x\|$. As the input moves far away from the training data, the behaviour is dominated by a single affine component whose largest logit can grow without bound, while all other logits remain comparatively smaller. Since the softmax function magnifies differences between logits, this causes the predicted probability for a single class to approach one.

Theorem 10.1 (cf. Theorem 3.1, Hein et al. [729]). *Suppose that $\mathbf{V}^l$ does not contain identical rows for all $l = 1, \dots, R$, then for almost any non-zero input x and any $\varepsilon > 0$, there exists a scaling factor $\alpha > 0$ such that for some class k ,*

$$\text{softmax}_k(f(\alpha x)) \geq 1 - \varepsilon,$$

and

$$\lim_{\alpha \rightarrow \infty} \text{softmax}_k(f(\alpha x)) = 1.$$

In other words, deterministic ReLU networks extrapolate with *increasing certainty* as we move away from training data, regardless of the lack of evidence.

10.1.2 Bayesian perspective as a counter measure

The problem identified in the previous section is an artefact directly caused by the use of point estimates. The natural way to solve it is to assign a probability measure over the weights θ instead of committing to a single value. As we will see, this naturally leads to increasing variance away from the data, which tempers the confidence after the softmax is applied.

The Laplace approximation provides a tractable route to introduce this behaviour into neural networks. To recall, we approximate for a given trained network with parameters θ_* (the maximum a posteriori estimate), the posterior over parameters by a Gaussian:

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_*, \Psi(\theta_*)^{-1}),$$

where Ψ describes the curvature of the negative log posterior evaluated at θ_* . With linearising the pushforward around θ_* , the logits become Gaussian as well (cf. Section 8.1):

$$z(x) \sim \mathcal{N}(\mu(x), k(x, x))$$

with

$$\mu(x) = f(x, \theta_*), \quad k(x, x) = J_\theta f(x, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x, \theta_*)^\top. \quad (10.1)$$

For ReLU networks, the components of $J_\theta f(x, \theta_*)$ grow at most linearly with $\|x\|$, so the logit variance $k(x, x)$ grows quadratically with $\|x\|$ (equivalently, the standard deviation grows linearly). This means that, far from the training set, the variance in each logit matches the mean's linear growth. We will now see that this leads to finite asymptotic confidence in each class, bounded away from 1 and 0.

The predictive class probabilities are obtained by marginalising over the Gaussian posterior

$$p(y = k|x, \mathcal{D}) = \mathbb{E}_{z \sim \mathcal{N}(\mu(x), k(x, x))} [\text{softmax}_k(z)] .$$

While exact evaluation is infeasible, accurate analytic approximations exist (cf. Section 8.5). A well-known approximation in the binary case is due to MacKay [1106]:

$$\mathbb{E}[\sigma(z)] \approx \sigma\left(\frac{\mu}{\sqrt{1 + \frac{\pi}{8}s^2}}\right),$$

where $z \sim \mathcal{N}(\mu, s^2)$ is the Gaussian logit with mean μ and variance $s^2 = k(x, x)$, and $\sigma(t) = 1/(1 + e^{-t})$ is the logistic sigmoid.

This solution was established by Kristiadi et al. [928] in their paper *Being Bayesian, Even Just A Bit, Fixes Overconfidence In ReLU Networks*, where *Even Just A Bit* refers to the observation that the following theorem holds even when only the weights of the last layer are treated probabilistically, and *even* if the covariance $\Psi(\theta_*)^{-1}$ in the Laplace approximation is chosen in an *arbitrary* way (say, equal to the identity $\mathbf{I}$).

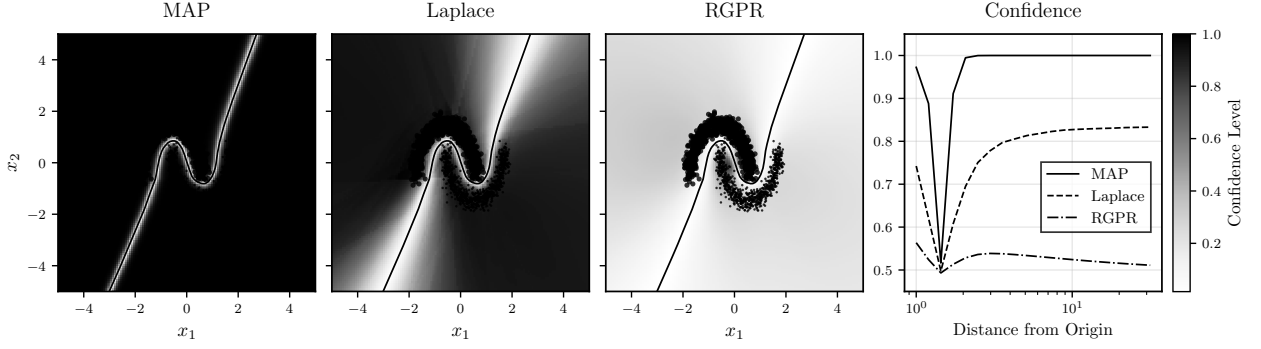


Figure 10.1: **Mitigating overconfidence in a ReLU network with the Laplace approximation.** The figure shows the predictive confidence on a two-moons binary classification task for three different models. Darker background shading indicates higher confidence, while the solid black line marks the decision boundary at $p = 0.5$. **(Left)** The standard MAP-trained ReLU network exhibits severe overconfidence. As described in Theorem 10.1, its predictive confidence incorrectly approaches 1 (maximum certainty) for inputs far away from the training data. **(Middle)** Applying the Laplace approximation introduces uncertainty over the network’s weights. The resulting logit variance grows quadratically with distance from the data, which tempers the softmax and reduces predictive confidence in out-of-distribution regions. **(Right)** Adding a ReLU-GP residual term (RGPR) to the Laplace-approximated logits introduces an additional source of variance that grows cubically. This ensures that the predictive confidence correctly converges to the uniform distribution (0.5) for inputs far from the training data, providing better calibration. **(Far Right)** This panel quantitatively confirms the visual results by plotting the maximum confidence of each model as an input point is scaled away from the origin.

Theorem 10.2 (cf. Proposition 2.5, Kristiadi et al. [928]). *Consider a binary classification ReLU network f_θ with Laplace posterior $\theta \sim \mathcal{N}(\theta_*, \Psi(\theta_*)^{-1})$. Then, the predictive confidence at input x decreases monotonically as prior variance σ_0^2 increases, satisfying*

$$\lim_{\sigma_0^2 \rightarrow \infty} \sigma(|z(x)|) \leq \sigma \left(\frac{|\mu(x)|}{1 + \sqrt{\frac{\pi}{8}} \lambda_{\max}(H) \|J_\theta f(x, \theta_*)\|^2} \right), \quad \lim_{\sigma_0^2 \rightarrow 0} \sigma(|z(x)|) = \sigma(|\mu(x)|).$$

Here, $\lambda_{\max}(H)$ denotes the largest eigenvalue of the negative log-likelihood Hessian evaluated at the posterior mean θ_* , while $\mu(x) = f(x, \theta_*)$ and $J_\theta f(x, \theta_*)$ are the mean logit and network Jacobian at θ_* , exactly as in Equation (10.1). Note that H differs from the posterior precision $\Psi(\theta_*)$ only by the Gaussian-prior term, $\Psi(\theta_*) = H + \sigma_0^{-2} \mathbf{I}$, and this prior contribution vanishes as $\sigma_0^2 \rightarrow \infty$, so only the likelihood Hessian survives in the bound.

Of course, this theorem provides only an asymptotic statement. In words: If we assign *any* Gaussian measure to the weights of the network (whether explicitly constructed as a Laplace approximation or simply imposed) then asymptotic overconfidence is “healed” in the weak sense that it is bounded away from 1 and 0. Despite its weak form, this result is a great argument for balancing the cost of Laplace approximations with their goals. If all you want is to fix asymptotic confidence, then you can do so at virtually zero computational cost: Just impose an arbitrary confidence on the parameters. This is still better than using a point estimate instead.

10.1.3 The role of the prior variance

Theorem 10.2 highlights the special role of the prior variance σ_0^2 : increasing σ_0^2 broadens the posterior over weights, which in turn accelerates variance growth in the logits and reduces predictive confidence away from the data. On the other hand, letting $\sigma_0^2 \rightarrow 0$ recovers the deterministic prediction. Although σ_0^2 is traditionally motivated by the weight decay used during training, in the Laplace setting it can be tuned *post hoc* to improve the calibration.

Hyperparameters such as the prior variance or additional factors entering the curvature can be tuned with techniques often motivated by the literature on Gaussian processes [1382]. This includes downstream objectives (e.g., negative log-likelihood, out-of-distribution entropy) or model fit criteria (e.g. log marginal likelihood). Here, the Gaussian structure of the Laplace approximation allows for closed-form objectives suitable for gradient-based tuning, which is discussed in more detail in addition to other applications in the next chapter (cf. Chapter 11).

For the particular setting of calibrating σ_0^2 such that the resulting uncertainty structure reflects out-of-

distribution well, Kristiadi et al. [928] suggest using

$$\operatorname{argmin}_{\sigma_0^2} -\frac{1}{n} \sum_{i=1}^n \log p(y = \hat{t}_i \mid \hat{x}_i, \mathcal{D}) - \gamma H[p(y \mid \tilde{x}_i, \mathcal{D})], \quad (10.2)$$

where the first term is the standard cross-entropy loss over an in-distribution validation set $\{\hat{x}_i, \hat{t}_i\}$, while the second is the predictive entropy over a set of out-of-distribution inputs $\{\tilde{x}_i\}$, which the objective encourages to be large through the leading minus sign. The parameter $\gamma \in [0, 1]$ controls the trade-off between these two terms.

10.1.4 How to mitigate the problem even further

A simplified summary of Theorem 10.2 is that assigning a Gaussian measure to weights bounds asymptotic confidence away from the pathological values 1 and 0. But it will not drive asymptotic confidence down to the ideal uniform (maximum entropy) limit $1/C$ where C is the number of classes. And of course the Theorem says nothing about confidence “close to” or “in-between” the training data. To address either problem, Kristiadi et al. [929, 930] consider two extensions to tune the overall confidence structure: one reshapes the local curvature for better calibration close to the data [930], the other combines the weight-space uncertainty with some additional output-space uncertainty to ensure the ideal uniform confidence limit [929].

The approach *Learnable Uncertainty under Laplace approximations* (LULA) [930] augments a trained network with additional parameters – *uncertainty units* – inserted into the architecture in a way that preserves the MAP prediction $f_{\theta_*}(x)$ but modifies the Hessian H , e.g. for three linear layers without nonlinearities and biases for simplicity, we have,

$$f(x) = \begin{bmatrix} W_{\text{MAP}}^{(3)} \end{bmatrix} \begin{bmatrix} W_{\text{MAP}}^{(2)} \end{bmatrix} \begin{bmatrix} W_{\text{MAP}}^{(1)} \end{bmatrix} x \quad (10.3)$$

$$\Rightarrow f^{(\text{LULA})}(x) = \begin{bmatrix} W_{\text{MAP}}^{(3)} & 0 \end{bmatrix} \begin{bmatrix} W_{\text{MAP}}^{(2)} & 0 \\ \hat{W}_1^{(2)} & \hat{W}_2^{(2)} \end{bmatrix} \begin{bmatrix} W_{\text{MAP}}^{(1)} \\ \hat{W}_1^{(1)} \end{bmatrix} x. \quad (10.4)$$

Therefore, the final variance changes while the mean prediction stays the same, which motivates the name *uncertainty units*. These units are then trained using an uncertainty-aware loss that encourages high predictive entropy on out-of-distribution (OOD) inputs and low entropy on in-distribution (ID) inputs, with gradients masked so that the original prediction remains unchanged. It can then be shown that the resulting Laplace approximation with diagonal curvature approximation has higher variance over the output compared to the diagonal Laplace for the network without the uncertainty units (cf. Proposition 2, Kristiadi et al. [930]).

However, residual overconfidence can still be left for inputs extremely far from the training data. This is a consequence of the fact that ReLU networks have only finitely many effective features, whereas a Gaussian process (GP) with an infinite set of ReLU features exhibits a growth of cubic variance, forcing the softmax confidence toward the uniform distribution as $\|x\| \rightarrow \infty$. The *ReLU-GP residual* (RGPR) approach [929] adds such a GP as an independent residual term $\hat{f} \sim \mathcal{GP}(0, \hat{k})$ to the Bayesian neural network:

$$\tilde{f}(x) = f(x) + \hat{f}(x)$$

with a specific double-sided cubic spline kernel $\hat{k}$ chosen so that $\hat{k}(x, x) \approx 0$ near the origin but grows cubically with $\|x\|$. The resulting logit distribution is

$$z(x) \sim \mathcal{N}(\mu(x), k(x, x) + \hat{k}(x, x) \mathbf{I})$$

preserving the mean while adding variance that dominates in the tails. Applying a softmax approximation to this Gaussian results in a predictive distribution that converges to the uniform distribution in the limit (cf. Theorem 4, Kristiadi et al. [929]).

Importantly, this Gaussian process term can be added in such a way that its evaluation has constant and negligible cost, independent of the training data. This is because it is explicitly constructed to not be affected by data within the training region, only contributing prior uncertainty in the OOD regime. It is thus another example of a Laplace approximation adding specific functionality, at extremely minimal cost overhead. The double-sided cubic spline kernel chosen in this work also has a pleasing interpretation as a limit process arising from an infinite number of the ReLU uncertainty units from above, all distributed in a regular fashion around the training domain, pointing “outwards” so that they are not affected by the training data. The implied intuition is that these units “complete” or complement the finitely many ReLUs that make up the deep network that we actually train on the

available (finite) data, and provide a nonparametric sea of ReLUs that cover the domain towards the horizon. To speak even more metaphorically, this is arguably an example of the Bayesian perspective on deep learning providing an intuition – and a concrete, practical solution! – to a pathology of point-estimated deep models. Figure 10.1 compares the predictive confidence of the MAP network, its Laplace approximation, and the RGPR extension on a two-moons task, illustrating how each successive step tempers overconfidence away from the data.

Takeaways

Asymptotic and OOD overconfidence are examples of pathologies arising from the use of point estimates in deep learning. The framework of Laplace approximations (or more generally, assigning a Gaussian measure to the weights and using linearisation to push forward to the output domain) can fix these issues to some tangible, quantifiable degree, while adding, occasionally negligible, or at least very manageable computational overhead. This is arguably because these problems really just arise discretely from a non-Bayesian treatment, and vanish as soon as one assigns *a* probability measure, *any* kind of posterior on the parameters.

10.2 Fixing catastrophic forgetting in continual learning

In *continual learning*, a single neural network is trained on a sequence of tasks

$$t = 1, \dots, T, \quad \text{with datasets } \mathcal{D}_t, \quad (10.5)$$

with the goal of maintaining good performance on earlier tasks. This setting is increasingly relevant in real-world deployments: modern large-scale models are trained once, but are then confronted with a stream of evolving data and objectives. A key obstacle to adaptation is *catastrophic forgetting*, the loss of previously acquired abilities when learning new tasks.

A naïve approach is to fine-tune the current parameters θ_t by minimising the loss on the next dataset $\mathcal{D}_{t+1}$:

$$\theta_{t+1}^* = \operatorname{argmin}_{\theta} \mathcal{L}(\theta; \mathcal{D}_{t+1}). \quad (10.6)$$

However, this discards all information about earlier tasks: parameters drift toward the new optimum for adapting to the new data $\mathcal{D}_{t+1}$, and end up erasing information by neglecting critical weights for previous performance.

10.2.1 A Bayesian solution

From a Bayesian perspective, training on $\mathcal{D}_{t+1}$ should not start from an uninformative prior. Instead, the posterior after task t should become the prior for task $t + 1$:

$$p(\theta_{t+1} \mid \mathcal{D}_{1:t+1}) \propto p(\mathcal{D}_{t+1} \mid \theta_{t+1}) p(\theta_t \mid \mathcal{D}_{1:t}), \quad (10.7)$$

instead of the original uninformative isotropic Gaussian $p(\theta)$ in form of weight decay. Taking the negative log turns this into a regularised optimisation problem, where the regulariser encodes knowledge from earlier tasks.

If $p(\theta_t \mid \mathcal{D}_{1:t})$ is approximated by a Gaussian $\mathcal{N}(\theta_t^*, \Psi_t^{-1})$ via the Laplace approximation [1522], the next-task objective becomes:

$$\mathcal{L}_{t+1}^{\text{reg}}(\theta) = \mathcal{L}(\theta; \mathcal{D}_{t+1}) + \frac{\lambda}{2} \|\theta - \theta_t^*\|_{\Psi_t}^2, \quad (10.8)$$

where $\|v\|_{\Psi_t}^2 := v^\top \Psi_t v$ denotes the squared Mahalanobis norm induced by the precision Ψ_t . The precision matrix Ψ_t determines how strongly each direction in parameter space is protected: large precision (high curvature) means a stiff penalty in that direction, since the loss on the previous task would change rapidly, while low precision allows for adaptation. This yields an *anisotropic* weight-space prior that selectively resists change where it would harm past performance. The regularisation strength λ controls the trade-off between retaining performance on past tasks and adapting to new ones by scaling the precision of the prior: higher values express greater confidence in past parameters, lower values signal greater uncertainty and openness to change.

10.2.2 Bayesian filtering in weight space

Starting from the previously motivated interpretation and following Sliwa et al. [1522], we can formalise continual learning as Bayesian state estimation in weight space:

$$\theta_0 \rightarrow \theta_1 \rightarrow \dots \rightarrow \theta_T \quad (10.9)$$

with a Gaussian *state-transition* model and a likelihood defined by the task loss

$$p(\theta_{t+1} \mid \theta_t) = \mathcal{N}(\theta_t, Q) \quad (\text{process model}) \quad (10.10)$$

$$p(D_t \mid \theta_t) \propto \exp\left(-\frac{1}{\tau} \mathcal{L}(\theta_t; D_t)\right), \quad (\text{observation model}). \quad (10.11)$$

Continual learning becomes in this interpretation a problem of tracking a moving target in weight space: the process model predicts where the target might go next, and the observation model corrects this prediction based on new evidence. The additional process noise covariance Q models the expected drift between tasks: large Q corresponds to high *plasticity* (parameters may change substantially), while small Q enforces *stability* (parameters remain close).

The Bayesian filtering recursion consists of:

1. *Predict step*: propagate the posterior through the process model to obtain a prior for θ_t , (potentially) inflating the uncertainty by Q .
2. *Update step*: incorporate the likelihood from $\mathcal{D}_t$, pulling the mean toward the new optimum and shrinking uncertainty according to curvature.

This is analogous to a Kalman filter in parameter space, but with a nonlinear, non-Gaussian observation model arising from the neural network loss. The Laplace approximation replaces this intractable term with a local Gaussian, making the update step analytically tractable within the filtering framework.

10.2.3 From EWC to OSLA to Lo-Fi

The state-space formulation provides a unifying Bayesian view of continual learning. Different methods in the literature can be seen as specific choices for (i) how the posterior covariance is represented and updated, and (ii) whether parameter drift between tasks is allowed. The Laplace approximation supplies the Gaussian form needed for tractable updates, but the quality and scalability of the approach depend on the curvature approximation used.

Elastic Weight Consolidation (EWC). Kirkpatrick et al. [897] proposed *elastic weight consolidation* (EWC), which constrains parameters to remain close to the previous optimum θ_t^* using the diagonal of the Fisher information metric F_t :

$$\mathcal{L} = \mathcal{L}(\theta; \mathcal{D}_{t+1}) + \frac{\lambda}{2} (\theta - \theta_t^*)^\top \text{diag}(F_t) (\theta - \theta_t^*). \quad (10.12)$$

This is equivalent to the proposed filtering framework when choosing $Q = 0$ and a diagonal precision. In the sequential setting, a separate quadratic penalty can be added for each of the previous tasks (multi-penalty), or equivalently, the Fisher matrices can be accumulated into a single continuously updated penalty. EWC is computationally efficient but lacks correlation between parameters.

Online Structured Laplace Approximation (OSLA). Ritter et al. [1401] extended EWC by replacing the diagonal Fisher with a *Kronecker-factored* (KFAC) curvature approximation (cf. Section 8.4), capturing intra-layer parameter correlations while remaining scalable. In the Bayesian filtering view, OSLA maintains a single Gaussian posterior over parameters and updates it *recursively* after each task:

$$\Psi_t = \lambda \text{KFAC}(F_t) + \Psi_{t-1}, \quad (10.13)$$

where F_t is again the Fisher information matrix computed with the current MAP estimate on task t . The running update replaces the need to store separate penalties for each task, as in multi-penalty EWC, and naturally implements a sequential Bayesian update with $Q = 0$ (no assumed drift). By incorporating inner-layer correlations, OSLA produces a more faithful posterior approximation and better retention over long sequences of tasks.

Low-Rank Laplace-Gaussian Filtering (Lo-Fi). Chang et al. [259] and Sliwa et al. [1522] generalised the Bayesian framework to arbitrary Q , enabling explicit modelling of expected drifts between tasks. They approximate the precision matrix using a low-rank approximation of the GGN (plus diagonal representing the initial prior), which makes both, prediction and update step, scalable while still modelling correlation between different layers.

In this general Bayesian filtering view, the concept of *smoothing* also becomes available. After the forward filtering pass, an optional backward pass refines each task-specific parameter estimate using information from

future tasks without revisiting their original data. The process can be compared to passing a message along a chain of messengers, where each messenger (task) has access only to the message (parameters) they just received and their own local observations; over time, the message can degrade (catastrophic forgetting). Smoothing allows later messengers to send clarifications or corrections back upstream, improving earlier versions without re-reading the original letters (data). In a deep learning context, this means earlier task models can be improved based on knowledge from later tasks, while preserving data privacy by never sharing or replaying the original datasets from other tasks.

Takeaways

The Laplace approximation arising from pre-training a network can be used as a log prior (a regulariser) on subsequent tasks. Doing so encourages the model to use “unused” parts of the parameter space to “store” new training information, and thus mitigates forgetting during continual learning. The underlying algebraic idea is that Bayes’ theorem for Gaussians is a closure: Conditioning a Gaussian prior (an ℓ_2 regulariser) on a (linear) Gaussian likelihood (a Laplace approximated training loss) yields a Gaussian posterior (a Laplace approximated trained network). Hence, updating a Gaussian *posterior* with new data is conceptually the same thing as updating a Gaussian *prior*. The only real challenge is to manage the resulting data-structures well, to keep computational cost manageable. But since Laplace maps this process onto Gaussians, these computational aspects are all mapped to numerical linear algebra, for which very powerful, well-studied algorithms are readily available.

10.3 Data-centric objectives

The Laplace approximation provides a tractable way to understand the relationship between a model and its training data, which, in turn, improves properties such as predictive calibration (cf. Section 10.1) and the ability to learn sequentially (cf. Section 10.2). The same model-data relationship is central to the fields studying optimal data selection. Rather than relying on random sampling, methods have been developed to strategically choose the most informative data to label (active learning) and the most efficient points to evaluate (Bayesian optimisation). A fundamentally Bayesian approach underpins these strategies, leveraging a model’s current beliefs to guide data acquisition. Also more nuanced data-centric questions exist, such as determining the most effective training order (curriculum learning), that have been studied from a Bayesian perspective with the help of the Laplace approximation [410]. However, here, we focus on active learning and Bayesian optimisation due to their foundational connection to Bayesian principles.

10.3.1 Active learning: selecting what to label next

The goal of **active learning** is to train an optimal model with a minimal data collection effort. For instance, training neural solvers for partial differential equations often requires a vast amount of training data generated by costly numerical simulations. Instead of random generation, active learning aims to strategically select the most informative data points to label, thereby achieving the same model accuracy with significantly less numerical simulations [1216].

From a Bayesian perspective, this selection process can be formalised as an optimisation problem guided by information theory. MacKay [1107] proposed framing active learning as a sequential design task where the goal is to choose the next data point that is expected to provide the maximal information gain about the model’s parameters θ . This gain is measured by the expected reduction in the entropy of the parameter posterior after observing a new data point (x_i, y_i) :

$$\mathcal{U}(x_i) = \mathbb{H}[p(\theta \mid \mathcal{D})] - \mathbb{E}_{y \sim p(y \mid x_i, \mathcal{D})}[\mathbb{H}[p(\theta \mid \mathcal{D}, x_i, y)]] \quad (10.14)$$

where $\mathbb{H}[p] = -\int p(u) \log p(u) du$ denotes the (differential) entropy of a distribution p ; for a discrete distribution, such as the class-label predictive $p(y \mid \cdot)$, the integral is replaced by the sum $-\sum_u p(u) \log p(u)$.

In general, computing this expected information gain is intractable, though the linearised Laplace approximation provides again a solution. By approximating the posterior $p(\theta \mid \mathcal{D})$ with a Gaussian, MacKay [1107] showed that the expected information gain from a potential new data point at input x is a monotonic function of the model’s current predictive variance at that location $s^2(x) = J_\theta f(x, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x, \theta_*)^\top$. This provides a principled justification for the intuitive and widely used uncertainty sampling heuristic: *to learn the most, one should query the points where the model is currently most uncertain* [1107].

The objective of maximising parameter information is now widely known as *Bayesian Active Learning by Disagreement* (BALD) [784]. The BALD score is a mathematically equivalent reformulation of the original objective:

$$\text{BALD}(x) = H[p(y \mid x, \mathcal{D})] - \mathbb{E}_{p(\theta \mid \mathcal{D})} H[p(y \mid x, \theta)]. \quad (10.15)$$

This version provides the powerful intuition that one should query points where the model is overall uncertain about its prediction (the first term), but where individual parameters θ from the posterior are confident in their potentially disagreeing predictions (the second term is low).

It has been observed, that BALD’s focus on parameter uncertainty can be misaligned with the goal of predictive performance, as it lacks an explicit notion of the relevant input data distribution. This can lead to the selection of obscure or irrelevant data points that offer little benefit to typical test input. To address these shortcomings, the *Expected Predictive Information Gain* (EPIG) [1180] was proposed, which measures the expected information a candidate label y at input x provides about the label y_* of a random target input x_* drawn from a target distribution $p_*(x_*)$. This objective is formalised as the expected reduction in predictive entropy on a random target point, averaged over all possible outcomes of the candidate query:

$$\text{EPIG}(x) = \mathbb{E}_{p_*(x_*)} [H[p(y_* \mid x_*)] - H[p(y_* \mid x_*, x, y)]]. \quad (10.16)$$

Both the BALD and EPIG objectives are agnostic to the method used for posterior approximation. By shifting the entropy computation to the lower-dimensional prediction space (BALD) or considering only the predictive entropy (EPIG), this information-theoretic approach lessens the reliance on a specific form of weight-space uncertainty, opening the door for various uncertainty quantification tools. The Laplace approximation remains a fast and tractable method for obtaining the predictive uncertainty needed to compute either score. Ultimately, however, the performance of these acquisition functions is strongly coupled to the quality of the resulting uncertainty estimates.

10.3.2 Bayesian optimisation: finding the optimal point

Many problems can be framed as finding the global optimum of an objective function that is very expensive to evaluate. Bayesian optimisation (BO) is a sequential strategy for finding the global optimum of such expensive black-box functions and works by combining a probabilistic surrogate model of the objective with acquisition functions to select the next point for evaluation [584]. Gaussian Processes (GPs) are the standard surrogate model because they are inherently Bayesian, providing a posterior distribution over the function and allowing for many popular acquisition functions to be computed analytically, making the entire process efficient and principled.

The linearised Laplace approximation (LLA) provides a way of using neural networks as surrogates by interpreting them as a Gaussian Process, which allows for reusing existing analytic acquisition functions, such as maximising the *upper confidence bound* given by

$$\alpha_{\text{UCB}}(x) = f(x, \theta_*) + \beta s(x) \quad (10.17)$$

where $f(x, \theta_*)$ is the network’s mean prediction at the MAP estimate, $s(x)$ is the predictive standard deviation from the LLA posterior and β a hyperparameter balancing exploration and exploitation [932].

The final optimisation process is then iterative. At each step, a neural network is trained on the existing data set and queried using the LLA and an acquisition function for the next data point to evaluate the original black-box function. Then, this new information must be incorporated by retraining or adapting the neural network on the augmented dataset to find a new maximum-a-posteriori (MAP) estimate and then re-computing the Laplace approximation around this new point. This refines the model’s beliefs and its weight-space uncertainty, guiding the next decision in the sequence.

Kristiadi et al. [932] demonstrated that the LLA surrogate inherits the strong predictive power and inductive biases of its underlying neural network, allowing it to outperform standard GPs on complex, high-dimensional tasks like searching through image spaces. However, they also note that broken predictive behaviour in extrapolation regions can significantly derail the optimisation process. This becomes clear when considering the Upper Confidence Bound as an example. For a ReLU network, the predictive mean outside the data region becomes a non-constant linear function. This means the predicted value will increase or decrease indefinitely as you move away from the data, while the uncertainty term also tends to grow linearly in these extrapolation regions, following the discussion in Section 10.1.2. Since both terms grow linearly, and the UCB score uses their sum, this utility also increases without bound in the same direction, leading the optimiser to choose a point infinitely far away in an unbounded search space. Note that this behaviour is consistent within the context of UCB: A region of extremely large expected reward *and* extremely large uncertainty is a very appealing target for another experiment. So the

problem here is arguably not the Bayesian treatment, but a badly chosen model class. Namely one that quite literally believes in pots of gold at the end of the rainbow.

Takeaways

Laplace approximations endow deep networks with probabilistic *uncertainty*, at tractable computational cost. A probability measure in itself, though, is just an intermediate mathematical notion. Its principal philosophical purpose, perhaps, is to allow meaningful *decisions*. For example, the Laplace posterior can be used to decide where to collect more training data in order to most efficiently reduce uncertainty and thus improve the model. Or, less directly, to find points where the target function (or a function of it) takes extremal values, provides most return. Interestingly, because these decisions have to be taken within a fixed domain, the Bayesian posterior used for this purpose does not have to be “correct” in an absolute sense, only a relative one. This highlights that when picking an approximate inference scheme for Bayesian deep learning, one should consider the downstream use case.

Chapter 11

Model selection with the marginal likelihood

The main intractability of high-dimensional Bayesian inference, as mentioned earlier in the book, stems from the difficulty of computing the normalising constant of the posterior distribution—the *marginal likelihood* (or *evidence*). While the MCMC methods presented before circumvent this problem by allowing us to sample from an *unnormalised* posterior distribution, the Laplace approximation offers the benefit that we can compute this quantity in closed form due to the Gaussianity. This not only enables inference and predictions (as shown in the previous chapters), but has one additional important use case: *model selection*. In this chapter, we will motivate why the marginal likelihood is a useful model selection criterion, explain how it is computed with the Laplace approximation, and present several practical use cases for model selection in Bayesian deep learning.

11.1 Marginal likelihood for model selection

There are several perspectives to understand why the marginal likelihood can be a useful criterion for model selection. We will review some different ones in this section to hopefully convince you that this pesky normalisation constant is not only a nuisance but a powerful tool when wielded in the right way.

11.1.1 Hierarchical Bayes

Let us look at the familiar Bayes’ rule, but with an explicit conditioning on our model of choice, which we will denote by $\mathcal{M}$. This model can encompass both the prior and the likelihood, so in the case of Bayesian deep learning, for instance, the neural network architecture, the likelihood noise level, the prior variance, or its distributional form. In short, the model $\mathcal{M}$ contains everything we have to choose actively before we can start to perform inference. The model-conditioned Bayes’ rule looks like this:

$$p(\theta \mid \mathcal{D}, \mathcal{M}) = \frac{p(\theta \mid \mathcal{M}) p(\mathcal{D} \mid \theta, \mathcal{M})}{p(\mathcal{D} \mid \mathcal{M})} \quad (11.1)$$

If we assume now that we do not actually know how to select the model in advance and would like to be Bayesian about it, we could attempt to treat $\mathcal{M}$ as a random variable in a hierarchical Bayesian model, the posterior of which would take the form

$$p(\mathcal{M} \mid \mathcal{D}) = \frac{p(\mathcal{M}) p(\mathcal{D} \mid \mathcal{M})}{\int p(\mathcal{M}) p(\mathcal{D} \mid \mathcal{M}) d\mathcal{M}} \quad (11.2)$$

If we further assume that *a priori*, we do not have any preference for any particular model, that is, $p(\mathcal{M})$ is uniform across models (brushing under the rug the fact that this may be an improper prior if the space of models is not integrable), this yields

$$p(\mathcal{M} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mathcal{M}) \quad (11.3)$$

We now see that the hierarchical posterior over models is proportional to the marginal likelihood term in Equation 11.1. Therefore, if we want to select the model that maximises the posterior (i.e., a MAP estimate over models), we just need to optimise $\mathcal{M}$ with respect to the marginal likelihood! Since this is essentially just performing a maximum likelihood selection on the second level of this Bayesian hierarchy, it is also often called *maximum likelihood II* (*ML-II*) model selection.

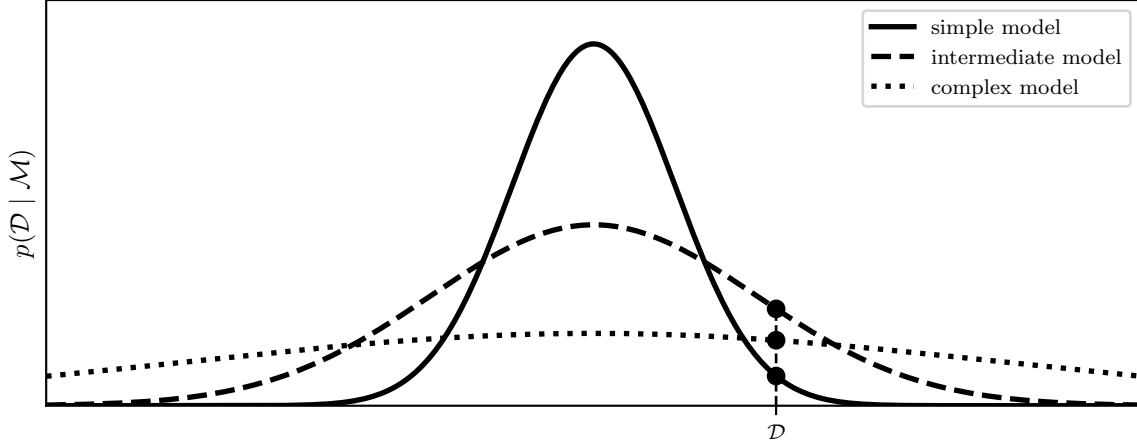


Figure 11.1: A visualisation of the Occam’s razor effect of the marginal likelihood. The simplest model can assign the largest likelihoods, but just to a small number of datasets. For the most complex model, it is the other way around. For a given dataset $\mathcal{D}$, the marginal likelihood selects an intermediate model that is just complex enough to model these data.

11.1.2 Occam’s razor

Apart from the interpretation as maximum likelihood in a hierarchical Bayesian model, we can also motivate from first principles why the marginal likelihood as a model selection criterion leads to an automatic *Occam’s razor* effect. The principle of Occam’s razor states that among a set of hypotheses (or models) that explain the same data equally well, one should always prefer the simplest one [1531]. The connection to the marginal likelihood has already been made in David MacKay’s popular textbook [1111], based on his thesis [1112], and has been further developed by Rasmussen and Ghahramani [1380].

The argument is relatively simple: since the term $p(\mathcal{D} | \mathcal{M})$ has to be a normalised distribution over all possible datasets $\mathcal{D}$ for any given model $\mathcal{M}$, this means that models can either assign a lot of likelihood to a small number of datasets or a small amount of likelihood to many datasets. The former would intuitively be a *simple* model, such as a linear one, whereas the latter would be a *complex* model, such as a large neural network. For a dataset that can be modelled well by a linear function, the linear model will therefore be able to assign a larger likelihood than the neural network, because its likelihood is not distributed over all the non-linear datasets. Generally, this means that the model with the largest marginal likelihood for a given dataset will typically be a model that is just complex enough to model this dataset, but not more complex, thus implementing Occam’s razor. This intuition is visualised in Figure 11.1.

11.1.3 Cross-validation

A popular non-Bayesian model selection criterion is cross-validation. The idea is that you split your dataset into $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}}$, with disjoint training and validation sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. We can then estimate the generalisation risk (i.e., the expected loss on unseen data) by training a model on $\mathcal{D}_{\text{train}}$ and then evaluating its error on $\mathcal{D}_{\text{val}}$. This fits the intuition that we are ultimately interested in applying our model to unseen test data $\mathcal{D}_{\text{test}}$, so we want to know how well the model predicts on datapoints it has not seen during training.

In their seminal paper, Fong and Holmes [537] show that the log marginal likelihood on $\mathcal{D}$ is equal to the average validation log likelihood over all possible cross-validation splits, that is,

$$\log p(\mathcal{D}) = \mathbb{E}_{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}} \sim \text{split}(\mathcal{D})} [\log \mathbb{E}_{\theta \sim p(\theta | \mathcal{D}_{\text{train}})} [p(\mathcal{D}_{\text{val}} | \theta)]] \quad (11.4)$$

where $\text{split}(\mathcal{D})$ is a shorthand for writing a mechanism that first uniformly samples a dataset size split $m < N$ and then samples a uniformly random permutation of assigning $N - m$ points to $\mathcal{D}_{\text{train}}$ and m points to $\mathcal{D}_{\text{val}}$. Note that for notational convenience, we have suppressed the explicit conditioning on $\mathcal{M}$, but you can indeed read all the $p(\cdot)$ here as implicitly being $p(\cdot | \mathcal{M})$.

Obviously, actually performing this exhaustive cross-validation procedure would require a computational cost scaling combinatorially with the dataset size N , so nobody would actually compute this in practice. However, we see that with the marginal likelihood, we get the result of this exhaustive procedure by solving just a single integral

over the training dataset. This is another argument for the marginal likelihood as a model selection criterion: if you believe that cross-validation is a good way to evaluate the potential generalisation capabilities of your model, the marginal likelihood will not only compute one cross-validation for you, but *all* of them at the same time!

11.1.4 PAC-Bayes

Another way to study the generalisation behaviour of Bayesian predictors is to use the framework of PAC-Bayes bounds [1154, 1507]. While we will get a more thorough introduction to PAC-Bayes bounds in later chapters (specifically in Chapter 22), for now it suffices to say that they are mathematical statements that bound the expected true risk of our (approximate) posterior—which we cannot know because we do not have access to the true data-generating process—with the expected empirical risk that we can actually measure. These bounds generically look something like this:

Proposition 11.1 (Generic PAC-Bayes bound [24]). *For any (approximate) posterior $q(\theta)$, choice of prior $p(\theta)$, and choice of $\delta \in (0, 1)$, the expected true risk is bounded with probability at least $1 - \delta$ over sampling of datasets $\mathcal{D}$ of size N from the data-generating process as*

$$\mathbb{E}_{\theta \sim q(\theta)}[\mathcal{L}(\theta)] \leq \mathbb{E}_{\theta \sim q(\theta)}[\hat{\mathcal{L}}(\theta, \mathcal{D})] + \mathcal{C}(q(\theta), p(\theta), N, \delta) \quad (11.5)$$

Here, $\mathcal{C}(q(\theta), p(\theta), N, \delta)$ is a term that measures the model complexity. In a simple case, this could, for instance, be [1154, 1314]

$$\mathcal{C}(q(\theta), p(\theta), N, \delta) = \sqrt{\frac{D_{\text{KL}}(q(\theta) \| p(\theta)) + \log \frac{2\sqrt{N}}{\delta}}{2N}} \quad (11.6)$$

This bound fits several key intuitions: it gets tighter as the size of the dataset grows because the model can overfit less; it gets looser if we reduce δ because then the bound needs to hold with higher probability; and it gets looser as the KL-divergence between posterior and prior grows because then the model can overfit more.

In summary, to get a small bound on the expected risk, you need a posterior that predicts well in terms of empirical risk and at the same time is as simple as possible in terms of the complexity measure $\mathcal{C}$. If this again sounds like Occam’s razor to you, you will not be surprised to learn that PAC-Bayes bounds can be formally connected to the marginal likelihood (indeed, this section would probably not be part of this chapter otherwise).

If we take the posterior in the bound to be the true Bayes posterior, Germain et al. [610] derive the following PAC-Bayes bound. Note that the true posterior is often intractable and can only be approximated, in which case the bound will only hold approximately. Note also that it only holds under the technical condition that the log likelihood is sub-Gamma; a sub-Gamma loss is one whose tails are upper-bounded by the tails of a Gamma distribution, essentially ensuring that the loss does not grow unexpectedly fast. This holds, for instance, for Bayesian linear regression with a Gaussian likelihood.

Proposition 11.2 (Marginal likelihood PAC-Bayes bound [610]). *For any choice of prior $p(\theta)$, and choice of $\delta \in (0, 1)$, and if the negative log likelihood loss $\ell(\theta, \mathcal{D}) = -\log p(\mathcal{D} | \theta)$ is sub-Gamma with variance factor s^2 and scale $c < 1$, the expected true risk under the Bayes posterior $p(\theta | \mathcal{D})$ is bounded with probability at least $1 - \delta$ over sampling of datasets $\mathcal{D}$ of size N from the data-generating process as*

$$\mathbb{E}_{\theta \sim p(\theta | \mathcal{D})}[\mathcal{L}(\theta)] \leq \frac{s^2}{2(1-c)} - \frac{1}{N} \log(\delta p(\mathcal{D})) \quad (11.7)$$

We therefore see that maximising the marginal likelihood directly minimises this bound on the generalisation error, providing another strong (frequentist) argument for why it is a reasonable criterion for model selection. Remember, however, as a caveat, that this result only strictly holds for sub-Gamma likelihoods, such as for linearised Laplace in the regression setting, but not in the case of full Laplace inference in BNNs, especially in the classification setting. For those cases, the bound does not provide a proper guarantee, but still an intuitive argument for why the Occam’s razor effect of the marginal likelihood can improve generalisation.

11.2 Differentiable marginal likelihood with Laplace

So now that we have convinced ourselves that the marginal likelihood is a great criterion for model selection, we are left with the problem of actually computing (or approximating) it. Indeed, this is one of the main challenges of Bayesian inference, which is why, in practice, the marginal likelihood is not often used for this purpose. However,

as promised above, the Laplace approximation has the lucky property that we can actually compute the marginal likelihood in closed form!

To understand how we can do that, we will need to look under the hood of the Laplace approximation in Equation 7.4b. Let us remind ourselves that we are aiming to approximate the posterior

$$p(\theta \mid \mathcal{D}) = \frac{p(\theta)p(\mathcal{D} \mid \theta)}{p(\mathcal{D})} = \frac{1}{p(\mathcal{D})} \tilde{p}(\theta) \quad (11.8)$$

where we have now written the unnormalised posterior as $\tilde{p}(\theta)$. In the context of model selection, feel free to mentally condition all of these distributions on $\mathcal{M}$ as in Equation 11.1 above. We will omit this conditioning here to not needlessly clutter the notation. Note also that we can choose the regularised risk from Equation 7.2 to be $\mathcal{L}_\Omega(\theta) = -\log \tilde{p}(\theta)$.

If we approximate $\log \tilde{p}(\theta)$ around a point θ_* , this gives

$$\log \tilde{p}(\theta) \approx \log \tilde{p}(\theta_*) + \nabla_{\theta_*} \log \tilde{p}(\theta_*)^\top (\theta - \theta_*) - \frac{1}{2}(\theta - \theta_*)^\top \Psi(\theta_*)(\theta - \theta_*) + \mathcal{O}(\theta^3) \quad (11.9)$$

where $\Psi(\theta)$ is the same Hessian operator as in Equation 7.5. Note that Ψ is the Hessian of $-\log \tilde{p}(\theta)$, hence the sign change in the Taylor expansion.

If we now choose θ_* to be a maximiser of the (log) posterior, such that the first-order term vanishes because $\nabla_\theta \log \tilde{p}(\theta) = 0$, and ignore the third and higher order terms, we can approximate $\tilde{p}(\theta)$ with a distribution $q(\theta)$, which we define as

$$q(\theta) = \frac{1}{Z_q} \exp\left(\log \tilde{p}(\theta_*) - \frac{1}{2}(\theta - \theta_*)^\top \Psi(\theta_*)(\theta - \theta_*)\right) \quad (11.10)$$

$$= \frac{\tilde{p}(\theta_*)}{Z_q} \exp\left(-\frac{1}{2}(\theta - \theta_*)^\top \Psi(\theta_*)(\theta - \theta_*)\right) \quad (11.11)$$

Now, instead of computing the true marginal likelihood $p(\mathcal{D}) = \int \tilde{p}(\theta) d\theta$, we can just approximate it with the normaliser of our approximate distribution, that is $p(\mathcal{D}) \approx q(\mathcal{D}) = Z_q$.

Since we see from the quadratic form in the exponential that $q(\theta)$ is multivariate Gaussian with mean θ_* and precision matrix $\Psi(\theta_*)$, we can look up the normalising constant, which is $\sqrt{(2\pi)^P \det \Psi(\theta_*)^{-1}}$, where P is the dimensionality of the parameter space. This means that $\tilde{p}(\theta_*)/Z_q = (2\pi)^{-P/2} \det \Psi(\theta_*)^{1/2}$. Therefore, our approximate log marginal likelihood is

$$\log Z_q = \log q(\mathcal{D}) = \log \tilde{p}(\theta_*) - \frac{1}{2} \log \det \Psi(\theta_*) + \frac{P}{2} \log 2\pi \quad (11.12)$$

$$= \log p(\theta_*) + \log p(\mathcal{D} \mid \theta_*) - \frac{1}{2} \log \det \Psi(\theta_*) + \text{const} \quad (11.13)$$

Note that in the last line, we assumed that the number of parameters P is the same across the different models $\mathcal{M}$ that are being compared. If the model selection is meant to be performed across models with different numbers of parameters, this term naturally is not constant and still needs to be considered in the marginal likelihood.

11.2.1 Computational considerations

Similar to the inversion Ψ^{-1} discussed in previous chapters, the determinant $\det \Psi$ also has a computational cost of $\mathcal{O}(P^3)$ if the matrix is of size $P \times P$. As discussed before, this can get very expensive when the dimensionality of the parameter space is large, that is, for large neural network models. However, luckily, we can use some computational tricks and approximations to reduce the computational cost, such as linearisation and the empirical Fisher information, as well as diagonal or low-rank approximations (see Chapter 8 and Chapter 40 for details). Since none of these considerations are unique to the marginal likelihood computation, we will not discuss them at length in this chapter.

11.2.2 Post-hoc vs. online model selection

When selecting models using the marginal likelihood, the most naïve way is to perform *post-hoc* model selection. For instance, we can train two models, $\mathcal{M}_0$ and $\mathcal{M}_1$, and then compute their respective (log) marginal likelihoods. For illustration purposes, let us say that $\mathcal{M}_0$ is a fully-connected neural network and $\mathcal{M}_1$ is a convolutional neural network. We train them both on some image dataset (e.g., MNIST) and then compute $\log q(\mathcal{D} \mid \mathcal{M}_0) - \log q(\mathcal{D} \mid$



Figure 11.2: Using the Laplace-approximated marginal likelihood for online model selection of the prior variances on a small Bayesian neural network on a 2D classification task. We see that the marginal likelihood converges to a local optimum over the course of training and that the optimised model is better regularised than a naïvely trained (overfitting) model. [Adapted from 812]

$\mathcal{M}_1$). If this quantity is positive, we select $\mathcal{M}_0$, if it is negative, we select $\mathcal{M}_1$. Note that this difference between log marginal likelihoods is equivalent to the likelihood ratio test, which, according to the famous Neyman-Pearson lemma [1233], is the uniformly most powerful hypothesis test.

This selection approach gives us all the benefits of the marginal likelihood described in the previous section, but it still requires that we train each of the different models we want to test. Indeed, for selecting some parts of the model, we can do much better than that! While the neural network architecture is a discrete choice that we have to test post hoc, there are hyperparameters of the neural network with respect to which we can actually compute gradients of the marginal likelihood and then optimise them *online*. These include, for instance, the variance(s) of the prior or the likelihood noise. We will denote as $\mathcal{M}^\partial$ the subset of parameters for which gradients $\nabla_{\mathcal{M}^\partial} \log q(\mathcal{D} | \mathcal{M})$ exist. We can then optimise these parameters by performing simple gradient ascent like

$$\mathcal{M}^\partial \leftarrow \mathcal{M}^\partial + \gamma \nabla_{\mathcal{M}^\partial} \log q(\mathcal{D} | \mathcal{M}) \quad (11.14)$$

where γ is a step size. This allows us to select all these model hyperparameters within a single training run, without the need to train several models, as would be the case for approaches such as cross-validation.

However, this approach comes with a few caveats: First, recall that the marginal likelihood depends on the Hessian $\Psi(\theta_*)$, so on the second derivatives of the loss. If we want to take gradients of the marginal likelihood, we need to ensure that these are differentiable with respect to the hyperparameters, which is not true for all activation functions. Particularly, it is not true for the ReLU activation function, so we need to ensure to use smoother ones if we want to perform online model selection (e.g., sigmoid, tanh, Swish, GeLU, ELU, etc.). Second, since the marginal likelihood depends on the entire dataset $\mathcal{D}$, computing it requires an entire pass over the dataset, so in deep-learning parlance, one epoch. Since the best parameters θ_* generally depend on the choice of $\mathcal{M}^\partial$, we typically interleave the training, that is, we train θ for one or more epochs and then make one or several gradient steps on the marginal likelihood as written above, each of which corresponds to an additional epoch on the data. Depending on the frequency of these updates, the training will therefore require more epochs; in practice, typically around twice as many. Finally, dropping the first-order term in the Taylor expansion above required θ_* to be a true optimum of the loss, where the gradients vanish, but one might wonder whether standard optimisers actually find such optima in deep learning. Indeed, in practice, the optima we find do not truly have zero gradients everywhere. However, it has been shown empirically that this generally does not impair the model selection with the marginal likelihood [812] and there are even some theoretical arguments for why this might be the case [1041].

11.2.3 Simple example

As a simple illustrative example of online model selection with the marginal likelihood, we will consider a small fully-connected neural network for a synthetic 2D classification task. We place a Gaussian prior over the weights and we assume that the prior variance is shared within each layer. That is, the i th weight in the ℓ th layer will have the prior $p(\theta_{\ell i}) = \mathcal{N}(0, \sigma_\ell^2)$. Our hyperparameters to select are now these layerwise variances, so $\mathcal{M}^\partial = \{\sigma_\ell^2\}_{\ell=1}^L$.

We see in Figure 11.2 that our estimate of the marginal likelihood increases monotonically over the course of training and seems to converge to a local optimum. We also see that the prior variances indeed change during training and also seem to converge. The final model achieves a lower training accuracy compared to a naïvely trained model, but a higher test accuracy. This indicates that our selected model is better regularised and overfits less, in line with the Occam’s razor effect of the marginal likelihood. The visual smoothness of its decision boundary supports this conclusion.

11.2.4 On the risk of overfitting

Based on the previous example and our arguments above, you may now be convinced that the Laplace marginal likelihood will always select the best model for you, no matter the circumstances. Unfortunately, you would be wrong about that. We do not want to hide the caveat that maximum marginal likelihood (ML-II)—much like standard maximum likelihood (ML-I)—carries the inherent risk of overfitting. In our previous examples, such as choosing between two model architectures or optimising layerwise prior variances, this has not been apparent because the number of models compared (or number of hyperparameters optimised) was small compared to both the number of parameters of the model and the number of datapoints. However, if we were to optimise a number of hyperparameters (or compare a number of models) larger than P and N , we may well overfit the model to the data.

In such cases, just like in standard maximum likelihood, one remedy is to make the hyperprior $p(\mathcal{M})$ in Equation 11.2 more informative and thus regularise the optimisation. We could, for instance, have a separate prior for each weight $p(\theta_{\ell i}) = \mathcal{N}(\mu_{\ell i}, \sigma_{\ell i}^2)$ that we optimise with the marginal likelihood, but then have strong hyperpriors $p(\mu)$ and $p(\sigma^2)$ that regularise these two parameters to some reasonable values unless the marginal likelihood really provides strong evidence for different ones. For a more in-depth discussion on the risks of overfitting with the marginal likelihood in model selection and the use of hyperpriors, see Ober et al. [1253].

Another remedy, proposed by Lotfi et al. [1081], is to compute the marginal likelihood not on the whole dataset, but only on a subset, similar to cross-validation. The authors call this the *conditional log marginal likelihood* (CLML). While this can in practice indeed reduce the overfitting effects, it inherits a lot of the computational challenges of standard cross-validation, most importantly the need to choose a validation split or to perform the procedure several times across different splits.

11.2.5 Connection to flat minima

A lot of the arguments we made above in Section 11.1 hold for the true marginal likelihood $p(\mathcal{D})$. Based on some of the arguments in Chapter 7, you might wonder whether our approximation $q(\mathcal{D})$ is not too crude for model selection purposes. If the true BNN posterior has exponentially many modes, each of which has potentially a non-Gaussian geometry, should we really believe that approximating the marginal likelihood with a single Gaussian on one mode is enough to judge the quality of the model? Indeed, the proof of the metaphorical pudding is in the eating, and we will see in the following section that the Laplace marginal likelihood can effectively be used for many model-selection use cases in practice. However, before we get to the empirics, there is one more conceptual argument we can offer to hopefully appease the sceptical reader.

If we have another look at our Laplace marginal likelihood criterion in Equation 11.13, we see that it has three parts: the prior, the likelihood, and the log determinant of the Hessian. That means that models will achieve a large marginal likelihood estimate if they are probable under the prior, fit the data well (in terms of the likelihood), and have a small determinant of the Hessian. The latter of these is clearly the most cryptic. What does a small determinant mean? Recall that we assume that we are at a local optimum, or have at least linearised our network or added a sufficiently strong prior, such that the Hessian is positive definite. That means that all its eigenvalues are positive. Since the determinant is the product of the eigenvalues, a small determinant means that the eigenvalues of the Hessian are small, which in turn means that the loss function (our negative log posterior) has low curvature in the region around θ_* —it is relatively flat.

The idea that such flat minima can lead to better generalisation has been around for a while [759]. In fact, it is quite intuitive, since it essentially means that small changes in θ_* do not lead to large changes in the loss, so even if we are a bit off from the true optimum and even if the loss landscape shifts a bit due to distribution shift in the test data, we will probably still achieve a relatively low test error. This intuition is also sketched in Figure 11.3. If this combination of train error (likelihood) and complexity measure (flatness) sounds like a PAC-Bayes bound to you, you are on the right track: flat minima have indeed been connected to PAC-Bayesian results [699]. Moreover, the deep learning community has embraced flatness-encouraging training criteria—similar to our marginal likelihood—for its purposes, such as the famous sharpness-aware minimisation [SAM, 544] or its Bayesian counterpart [bSAM, 1197].

11.3 Use cases

As we have seen above, the Laplace marginal likelihood can be effectively used to select differentiable hyperparameters online, such as the prior variances or the likelihood noise, and to compare non-differentiable design choices post hoc, such as neural network architectures. These capabilities have been discovered by Immer et al. [812]

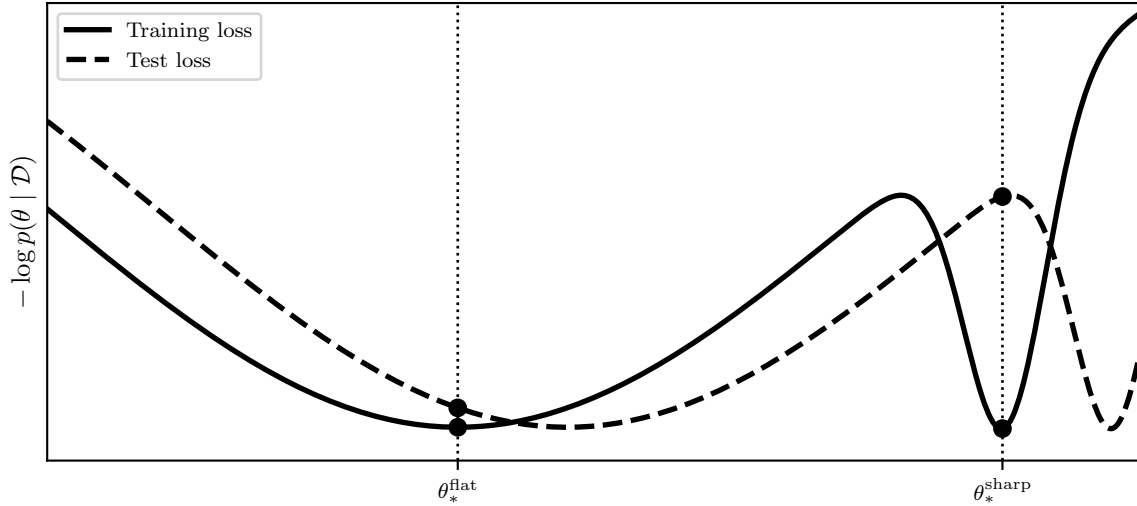


Figure 11.3: Here we show a loss landscape with a flat and a narrow minimum. As the loss landscape shifts from $\mathcal{D}_{\text{train}}$ to $\mathcal{D}_{\text{test}}$, the solution θ_*^{flat} still achieves a good performance, whereas the solution θ_*^{sharp} does not.

for modern deep learning models, and they have shown that this type of model selection is often cheaper and more effective than cross-validation. However, this type of hyperparameter tuning is by far not the only way the marginal likelihood can be used for model selection in deep learning. In recent years, different use cases have been developed, some of which have the potential to unlock entirely new application areas. While we cannot possibly provide a comprehensive overview of all of them, we will introduce a selection of the most interesting and creative ones in this section.

11.3.1 Feature selection

A common use case for model selection in Gaussian process models is feature selection via the automatic relevance determination (ARD) kernel [1382, 1736] (see also Chapter 23). This is a kernel in which each feature dimension has its own associated kernel length scale. These length scales can then be tuned using the marginal likelihood. One can perform a similar kind of feature selection in BNNs with the Laplace approximation, by performing model selection on the input weights. This is naturally more reasonable for data where the different input features have distinct semantic meaning, such as tabular data, than where they do not, such as images. One can even extend this idea to neural additive models (NAMs) [8]—that is, models where each input feature is passed to a separate neural network module—by performing model selection on the entire modules. Bouchiat et al. [197] have shown that the Laplace marginal likelihood provides an effective way to select features in Bayesian NAMs and even further improve their interpretability through its Occam’s razor effect.

11.3.2 Neural network pruning

The aforementioned feature selection, in its simplest implementation, amounts to removing weights from the input layer. This suggests the idea that one might also be able to remove weights from deeper layers. Indeed, this is the basic premise of the research field of neural network pruning [286]. There are two major approaches to pruning: unstructured pruning, where weights across the whole weight matrix are removed, and structured pruning, where entire rows or columns of the weight matrix are removed (this corresponds to removing neurons from the layer). Note that while the latter approach is more aggressive, it is the only one that immediately leads to computational gains, since the former approach just makes the matrix more sparse, but not smaller. Dhahri et al. [421] have shown that the Laplace marginal likelihood can be used to improve pruning in both settings, through its Occam’s razor effect, which encourages the neural networks to only use just as many weights as are needed for a given task. They have also shown that the Laplace posterior offers a simple yet effective pruning criterion: the log posterior density evaluated at 0.

11.3.3 Invariance learning

While the previous two use cases are still intuitively connected to our original model selection examples, as well as to existing applications in other models such as Gaussian processes, Immer et al. [815] propose to use the marginal likelihood for a more creative task: learning invariances from data. While invariances are commonplace in the context of designing invariant neural network architectures by hand (e.g., convolutional neural networks), there are many applications where invariances may be present in the data but the user does not know them. Following the approach from van der Wilk et al. [1662], Immer et al. [815] suggest using the following parameterisation of a generic invariant function:

$$\hat{f}(x, \theta, \eta) = \mathbb{E}_{x' \sim p(x' | x, \eta)}[f(x', \theta)] \quad (11.15)$$

Here, $f(\cdot)$ is the neural network, $\hat{f}(\cdot)$ is the desired invariant function, $p(x' | x, \eta)$ is a data augmentation distribution, with respect to which the function should be invariant, and η are hyperparameters controlling this augmentation distribution. The Laplace marginal likelihood can now be used to select η online during training and therefore enables learning a function that discovers the invariances directly from the data within a single training run. Immer et al. [815] show that this can faithfully recover known invariances and can drastically improve data efficiency on real datasets.

11.3.4 Probing

In addition to the bona fide model selection use cases introduced above, finally, the marginal likelihood can even be used for applications that—on the surface—do not look like model selection, such as probing. Probing is a common interpretability tool in language models, and it tries to answer the question of whether the latent representations of these models encode certain properties of their inputs. For instance, people might wonder “Does BERT encode the concept of verb tense?”. To answer this, there is a long line of work trying to take these latent encodings and fitting different simple readout functions to them, such as linear functions, to predict the property of interest, in this case, verb tense [109]. But what if the encodings do not encode the property linearly but quadratically? Then the linear probe might not perform well, but arguably, the property is still encoded. However, a quadratic model is more complex than a linear one, so maybe one should count this property as less well encoded than in a set of encodings where a linear probe would suffice. That again sounds suspiciously like Occam’s razor, and indeed Immer et al. [814] have shown that one can treat the problem of *jointly* selecting encodings and probes to predict the desired property as a model selection problem and use the Laplace marginal likelihood for a fair apples-to-apples comparison between different models. This shows that the tools presented in this chapter are much more widely applicable than one might naïvely expect, when one is a bit more creative in the interpretation of what *model selection* might look like.

Part III

Variational inference

Chapter 12

Introduction to variational inference

In probabilistic machine learning, our goal is often not just to find a single “best” set of parameters for a model, but to quantify our uncertainty over all possible parameters. We accomplish this using Bayesian inference.

Let $\Theta \subseteq \mathbb{R}^P$ denote the parameter space, and let $\theta \in \Theta$ be the vector of all parameters in our model (e.g., the weights and biases of a neural network). Let $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ be our observed training data, consisting of N input–target pairs. Bayesian inference updates our prior beliefs about the parameters, $p(\theta)$, after observing the data, yielding the *posterior* distribution $p(\theta | \mathcal{D})$.

This update is governed by Bayes’ theorem:

$$p(\theta | \mathcal{D}) = \frac{p(\mathcal{D} | \theta)p(\theta)}{p(\mathcal{D})}. \quad (12.1)$$

Here,

- $p(\theta | \mathcal{D})$ is the **posterior distribution**, summarising our updated beliefs about θ given the data;
- $p(\mathcal{D} | \theta)$ is the **likelihood**, viewed as a function of θ : for a given choice of parameters, it quantifies how likely the observed data are under the model;
- $p(\theta)$ is the **prior distribution**, which encodes our beliefs about θ before observing any data; and
- $p(\mathcal{D}) = \int p(\mathcal{D} | \theta)p(\theta) d\theta$ is the **marginal likelihood**, also called the **evidence**.

The central computational challenge in most modern Bayesian inference problems is evaluating $p(\mathcal{D})$, which requires integrating the joint distribution $p(\mathcal{D}, \theta)$ over the parameter space Θ . For models such as Bayesian neural networks, this integral is high-dimensional and generally lacks an analytic closed-form solution. Consequently, the posterior density $p(\theta | \mathcal{D})$ usually cannot be evaluated in normalised form, motivating approximate inference methods. This chapter focuses on variational inference, also known as variational Bayes, a widely used family of scalable approximate inference methods [170, 851, 1691].

12.1 A Variational Approach: Bayesian Inference as Optimisation

The core idea of variational inference (VI) is to reframe Bayesian inference as an optimisation problem. Instead of representing the generally intractable posterior $p(\theta | \mathcal{D})$ exactly, we choose a tractable distribution $q(\theta; \lambda)$ from a variational family $\mathcal{Q}$, where λ denotes its variational parameters. Concretely, $\mathcal{Q}$ could be a family of Gaussian distributions or a family of factorised distributions, as developed in Chapter 13. The goal is to find the member $q(\theta; \lambda^*)$ that best approximates the posterior.

12.1.1 The Kullback–Leibler Divergence

Following Kullback and Leibler [945], the **Kullback–Leibler (KL) divergence** from $q(\theta; \lambda)$ to the posterior $p(\theta | \mathcal{D})$ is

$$\begin{aligned} D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D})) &= \int q(\theta; \lambda) \log \frac{q(\theta; \lambda)}{p(\theta | \mathcal{D})} d\theta \\ &= \mathbb{E}_{q(\theta; \lambda)}[\log q(\theta; \lambda) - \log p(\theta | \mathcal{D})]. \end{aligned} \quad (12.2)$$

Because the KL divergence is asymmetric, its direction must be stated explicitly. VI seeks variational parameters satisfying

$$\lambda^* \in \arg \min_{\lambda} D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D})). \quad (12.3)$$

12.1.2 Rewriting the KL Objective

Expanding the posterior in Equation (12.2) using Bayes' theorem gives

$$D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D})) = \mathbb{E}_{q(\theta; \lambda)} [\log q(\theta; \lambda) - \log p(\mathcal{D}, \theta)] + \log p(\mathcal{D}). \quad (12.4)$$

Although the KL divergence itself cannot generally be evaluated exactly because $\log p(\mathcal{D})$ is unavailable, the log-evidence term is constant with respect to λ and therefore does not affect the optimiser in Equation (12.3). The remaining expectation does not require evaluating $p(\mathcal{D})$ and can be estimated using samples from $q(\theta; \lambda)$ whenever the joint density $p(\mathcal{D}, \theta)$ and variational density $q(\theta; \lambda)$ can be evaluated.

12.1.3 The Evidence Lower Bound (ELBO)

Rearranging Equation (12.4) yields

$$\log p(\mathcal{D}) = \underbrace{\mathbb{E}_{q(\theta; \lambda)} [\log p(\mathcal{D}, \theta) - \log q(\theta; \lambda)]}_{\mathcal{L}(\lambda)} + \underbrace{D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D}))}_{\geq 0}. \quad (12.5)$$

The first term defines the **evidence lower bound (ELBO)** $\mathcal{L}(\lambda)$. Because the KL divergence is non-negative,

$$\mathcal{L}(\lambda) \leq \log p(\mathcal{D}). \quad (12.6)$$

The gap between the ELBO and the log evidence is exactly $D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D}))$; the closer the variational distribution is to the posterior in this divergence, the smaller the gap. Since $\log p(\mathcal{D})$ is constant with respect to λ , Equation (12.5) also gives

$$\arg \max_{\lambda} \mathcal{L}(\lambda) = \arg \min_{\lambda} D_{\text{KL}}(q(\theta; \lambda) \| p(\theta | \mathcal{D})). \quad (12.7)$$

Thus, VI replaces evaluation of the posterior in normalised form with optimisation of an objective that can be estimated under suitable choices of likelihood, prior, and variational family.

12.1.4 Interpreting the ELBO

Using $p(\mathcal{D}, \theta) = p(\mathcal{D} | \theta)p(\theta)$, the ELBO can be written as

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\theta; \lambda)} [\log p(\mathcal{D} | \theta)] - D_{\text{KL}}(q(\theta; \lambda) \| p(\theta)). \quad (12.8)$$

The alternative form in Equation (12.8) reveals a trade-off between two terms:

- **Expected log-likelihood (data fit):** This term rewards variational distributions that place more probability in regions of parameter space where the observed data have high likelihood.
- **KL penalty (regularisation):** This term penalises departures of $q(\theta; \lambda)$ from the prior $p(\theta)$ and can therefore be viewed as a complexity penalty.

VI selects the member of $\mathcal{Q}$ that balances fit to the observed data against departure from the prior.

12.1.5 Other Lower Bounds on the Log-Evidence

The ELBO is one lower bound on $\log p(\mathcal{D})$. More generally, strictly positive unbiased estimators of $p(\mathcal{D})$ can induce lower bounds through Jensen's inequality; this optional subsection states the construction underlying importance-weighted and sequential Monte Carlo objectives.

Assume that $q(\theta; \lambda) > 0$ whenever $p(\mathcal{D}, \theta) > 0$. Then

$$\mathbb{E}_{q(\theta; \lambda)} \left[\frac{p(\mathcal{D}, \theta)}{q(\theta; \lambda)} \right] = \int p(\mathcal{D}, \theta) d\theta = p(\mathcal{D}). \quad (12.9)$$

For a concave function φ and random variable X , Jensen's inequality states that $\varphi(\mathbb{E}[X]) \geq \mathbb{E}[\varphi(X)]$. Taking $\varphi = \log$ in Equation (12.9) gives

$$\begin{aligned}\mathcal{L}(\lambda) &= \mathbb{E}_{q(\theta; \lambda)} \left[\log \frac{p(\mathcal{D}, \theta)}{q(\theta; \lambda)} \right] \\ &\leq \log \mathbb{E}_{q(\theta; \lambda)} \left[\frac{p(\mathcal{D}, \theta)}{q(\theta; \lambda)} \right] = \log p(\mathcal{D}).\end{aligned}\tag{12.10}$$

More generally, let $r(\mathbf{u}; \lambda)$ denote the sampling density of all random variables $\mathbf{u}$ used to construct a strictly positive unbiased estimator $g(\mathcal{D}; \mathbf{u}, \lambda)$ of $p(\mathcal{D})$, and suppose that $\mathbb{E}_{r(\mathbf{u}; \lambda)}[|\log g(\mathcal{D}; \mathbf{u}, \lambda)|] < \infty$. Then

$$\mathbb{E}_{r(\mathbf{u}; \lambda)}[g(\mathcal{D}; \mathbf{u}, \lambda)] = p(\mathcal{D}).\tag{12.11}$$

Jensen's inequality gives the lower bound

$$\mathcal{L}_g(\lambda) := \mathbb{E}_{r(\mathbf{u}; \lambda)}[\log g(\mathcal{D}; \mathbf{u}, \lambda)] \leq \log p(\mathcal{D}).\tag{12.12}$$

The standard ELBO is recovered by taking $\mathbf{u} = \theta$, $r(\theta; \lambda) = q(\theta; \lambda)$, and $g(\mathcal{D}; \theta, \lambda) = p(\mathcal{D}, \theta)/q(\theta; \lambda)$. For example, Burda et al. [220] introduce the importance-weighted autoencoder (IWAE) objective, which uses s independent samples $\theta_j \sim q(\cdot; \lambda)$ and the estimator

$$g(\mathcal{D}; \mathbf{u}, \lambda) = \frac{1}{s} \sum_{j=1}^s \frac{p(\mathcal{D}, \theta_j)}{q(\theta_j; \lambda)}, \quad \mathbf{u} = \{\theta_j\}_{j=1}^s.\tag{12.13}$$

Related lower bounds can be constructed using sequential Monte Carlo [1114]. Although $\mathcal{L}_g(\lambda)$ may not be available in closed form, it can often be optimised using Monte Carlo gradient estimators. However, a tighter bound alone does not guarantee better optimisation or predictive performance, because the signal-to-noise ratio and other properties of the associated gradient estimator also matter [1368].

In summary, VI approximates $p(\theta \mid \mathcal{D})$ by optimising a distribution from a tractable family, with the ELBO providing an objective that balances data fit against departure from the prior. Chapter 13 develops the mean-field construction in detail, and subsequent chapters consider further variational families and optimisation methods.

Chapter 13

Gaussian mean-field variational inference

In Chapter 12, we introduced the evidence lower bound (ELBO), a tractable objective that reframes Bayesian inference as an optimisation problem. However, optimising

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\theta; \lambda)}[\log p(\mathcal{D} \mid \theta)] - \text{D}_{\text{KL}}(q(\theta; \lambda) \parallel p(\theta)) \quad (13.1)$$

still requires evaluating an expectation. Unless the likelihood, prior, and variational family have special conjugate or linear-Gaussian structure, this expectation is rarely available in closed form and is generally unavailable for neural-network likelihoods.

To simplify optimisation, we adopt a strong structural assumption known as the *mean-field* assumption: the variational distribution factorises over the parameters, or over predefined groups of parameters. For $\theta = (\theta_1, \dots, \theta_P)$, the fully factorised form is

$$q(\theta; \lambda) = \prod_{i=1}^P q_i(\theta_i; \lambda_i). \quad (13.2)$$

Thus, the coordinates are independent under the approximation q , even though they are generally dependent under the true posterior $p(\theta \mid \mathcal{D})$. This assumption permits coordinate-wise updates, but it prevents q from representing posterior correlations and often leads to underestimated marginal posterior variances.

13.1 Coordinate Ascent Variational Inference

Under the mean-field assumption, the ELBO can be optimised by cyclically updating one factor q_j while holding the remaining factors fixed. This procedure is known as coordinate ascent variational inference (CAVI). Define

$$q_{-j}(\theta_{-j}; \lambda_{-j}) := \prod_{i \neq j} q_i(\theta_i; \lambda_i). \quad (13.3)$$

Substituting the factorised approximation into the ELBO and retaining only terms that depend on q_j gives

$$\log q_j^*(\theta_j) = \mathbb{E}_{q_{-j}(\theta_{-j}; \lambda_{-j})}[\log p(\mathcal{D}, \theta)] + \text{const}, \quad (13.4)$$

where the additive constant normalises q_j^* . Equivalently, the optimal factor is proportional to the exponentiated expected log-joint, with the expectation taken over the current approximation to all parameters other than θ_j .

The full procedure is given in Algorithm 17. Each exact coordinate update leaves the ELBO unchanged or increases it; under standard regularity conditions, repeated updates converge to a stationary point, but not necessarily to the global optimum. Closed-form CAVI updates are most readily available in conditionally conjugate models; otherwise, the expectation or the resulting update may itself require approximation.

For a recent hybrid approach combining variational inference and MCMC, see Alexos et al. [21].

We now strengthen the mean-field assumption by taking each factor to be Gaussian:

$$q_i(\theta_i; \lambda_i) = \mathcal{N}(\theta_i \mid \mu_i, \sigma_i^2), \quad \sigma_i > 0. \quad (13.5)$$

This is the *Gaussian mean-field* assumption. The variational parameters determine the per-coordinate means and positive standard deviations,

$$\lambda = \{(\mu_i, \sigma_i)\}_{i=1}^P. \quad (13.6)$$

Algorithm 17: Coordinate ascent variational inference (CAVI).

Result: Converged factors $q_1(\theta_1), \dots, q_P(\theta_P)$
Initialise all factors $q_i(\theta_i; \lambda_i)$ for $i = 1, \dots, P$
while the ELBO has not converged **do**
 for $j = 1, \dots, P$ **do**
 Update q_j using the CAVI rule:

$$q_j(\theta_j) \propto \exp(\mathbb{E}_{q_{-j}(\theta_{-j}; \lambda_{-j})}[\log p(\mathcal{D}, \theta)])$$

 end
end
return converged factors $q_1(\theta_1), \dots, q_P(\theta_P)$

For gradient-based optimisation, each positive standard deviation is typically represented through an unconstrained scale parameter. The variational parameters may be optimised by CAVI when closed-form updates are available or, more generally, by the stochastic-gradient methods introduced next.

13.2 Stochastic Gradient Optimisation

When closed-form CAVI updates are unavailable, the ELBO can instead be maximised by stochastic gradient ascent. Writing

$$g(\theta; \lambda) := \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta; \lambda), \quad (13.7)$$

we require an estimator of

$$\nabla_{\lambda} \mathcal{L}(\lambda) = \nabla_{\lambda} \mathbb{E}_{q(\theta; \lambda)}[g(\theta; \lambda)]. \quad (13.8)$$

The dependence of the sampling distribution $q(\theta; \lambda)$ on λ prevents a naive interchange of the gradient and expectation.

13.2.1 The Reparameterisation Gradient Estimator

A standard low-variance approach is the *reparameterisation gradient estimator*, also called the pathwise estimator [892, 1391]. For the Gaussian mean-field family, draw parameter-free noise and transform it according to

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \theta = \mu + \sigma \odot \epsilon. \quad (13.9)$$

This gives

$$\mathbb{E}_{q(\theta; \lambda)}[g(\theta; \lambda)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[g(\theta(\lambda, \epsilon); \lambda)]. \quad (13.10)$$

Under the usual regularity conditions, differentiation can now be moved inside the expectation:

$$\nabla_{\lambda} \mathbb{E}_{q(\theta; \lambda)}[g(\theta; \lambda)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\nabla_{\lambda} g(\theta(\lambda, \epsilon); \lambda)]. \quad (13.11)$$

Here ∇_{λ} is the total derivative through both $\theta(\lambda, \epsilon)$ and the explicit λ -dependence of g . Relative to score-function (REINFORCE) estimators, the reparameterisation estimator is typically much lower variance while remaining unbiased under the stated regularity conditions. With mini-batches, the gradient is doubly stochastic because of both data subsampling and the draw of ϵ ; larger mini-batches, multiple independent noise samples, and control variates can reduce variance at additional computational cost, while the resulting noise must be accounted for when selecting the step-size schedule. The pathwise construction also applies to other differentiable reparameterisable variational families.

Algorithm 18 gives a single-sample stochastic-gradient step.

13.3 Examples of Gaussian Mean-Field Variational Inference

We begin with a conjugate linear model, then turn to Bayesian neural networks and variational autoencoders.

Algorithm 18: Reparameterisation-gradient step for Gaussian mean-field variational inference.

Input: Variational parameters λ determining $\boldsymbol{\mu}$ and positive $\boldsymbol{\sigma}$, learning rate ρ , mini-batch size b
Result: Updated variational parameters λ

 Sample a mini-batch $\mathcal{B} \subseteq \{1, \dots, N\}$ of size b uniformly without replacement

 Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

 Compute $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$

/* Reparameterisation */

Compute the scaled mini-batch log-likelihood

$$\widehat{\ell}_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{N}{b} \sum_{n \in \mathcal{B}} \log p(y_n | x_n, \boldsymbol{\theta})$$

Compute the gradient estimate

$$\widehat{g} = \nabla_{\lambda} \left[\widehat{\ell}_{\mathcal{B}}(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}; \lambda) \right]$$

 Update $\lambda \leftarrow \lambda + \rho \widehat{g}$
return λ

13.3.1 Example 1: Bayesian Linear Regression

Consider the Bayesian linear regression model

$$y_n = \Phi(x_n)^\top \mathbf{w} + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \beta^{-1}), \quad (13.12)$$

 where β is the noise precision, $\Phi(\cdot)$ is a feature map, and the model parameters are $\boldsymbol{\theta} = \mathbf{w}$. We use

- **Prior:** $p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$;
- **Likelihood:** $p(y_n | x_n, \mathbf{w}) = \mathcal{N}(y_n | \Phi(x_n)^\top \mathbf{w}, \beta^{-1})$.

The exact posterior is a full-covariance Gaussian, which we write as

$$p(\mathbf{w} | \mathcal{D}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{S}). \quad (13.13)$$

We approximate it with the Gaussian mean-field family

$$q(\mathbf{w}; \lambda) = \prod_{i=1}^P \mathcal{N}(w_i | \mu_i, \sigma_i^2). \quad (13.14)$$

In this conjugate setting, the coordinate updates are available in closed form. More precisely, the reverse-KL-optimal diagonal Gaussian has $\mu_i = m_i$ and $\sigma_i^2 = 1/(\mathbf{S}^{-1})_{ii}$. Thus, it discards the off-diagonal covariance terms, and $\sigma_i^2 \leq S_{ii}$ shows explicitly how the mean-field approximation can underestimate the exact marginal variances when posterior correlations are present.

13.3.2 Example 2: Bayesian Neural Networks (BNNs)

 Consider a Bayesian neural network whose parameters $\boldsymbol{\theta}$ collect all weights and biases; the parameter count P may be very large. The model is

- **Prior:** $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} | \mathbf{0}, \mathbf{I})$ (other prior choices are possible);
- **Likelihood:** $p(\mathcal{D} | \boldsymbol{\theta}) = \prod_{n=1}^N p(y_n | x_n, \boldsymbol{\theta})$.

 With the Gaussian mean-field approximation $q(\boldsymbol{\theta}; \lambda) = \prod_{i=1}^P \mathcal{N}(\theta_i | \mu_i, \sigma_i^2)$, the ELBO is

$$\mathcal{L}(\lambda) = \sum_{n=1}^N \mathbb{E}_{q(\boldsymbol{\theta}; \lambda)} [\log p(y_n | x_n, \boldsymbol{\theta})] - \text{D}_{\text{KL}}(q(\boldsymbol{\theta}; \lambda) \| p(\boldsymbol{\theta})). \quad (13.15)$$

Because of the network's non-linear transformations, the expected log-likelihood is generally unavailable in closed form, so the analytic CAVI updates above do not apply. The ELBO can instead be optimised using the reparameterisation-gradient procedure in Algorithm 18, yielding the Bayes by Backprop approach [174]. The same stochastic-gradient construction applies to any differentially reparameterisable variational family, not only to Gaussian mean-field approximations.

13.3.3 Example 3: Variational Autoencoders

Variational inference also applies to generative latent-variable models. In a variational autoencoder (VAE) [892, 1391], VI approximates the posterior of a *local latent variable* $\mathbf{z}_n$ for each data point x_n , rather than a posterior over the decoder’s global parameters. The generative model is

1. **Prior:** $\mathbf{z} \sim p(\mathbf{z})$, for example $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$;
2. **Likelihood:** $p_\psi(x \mid \mathbf{z})$, represented by a decoder network with parameters ψ .

The goal is to maximise the marginal log-likelihood

$$\log p_\psi(x) = \log \int p_\psi(x \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}, \quad (13.16)$$

which is generally intractable. For any distribution $q(\mathbf{z})$, the variational principle from Section 12.1.3 gives

$$\log p_\psi(x) \geq \mathcal{L}(q, \psi; x) := \mathbb{E}_{q(\mathbf{z})}[\log p_\psi(x \mid \mathbf{z})] - \text{D}_{\text{KL}}(q(\mathbf{z}) \parallel p(\mathbf{z})). \quad (13.17)$$

The bound is tight at the true posterior $q(\mathbf{z}) = p_\psi(\mathbf{z} \mid x)$, but this posterior cannot generally be evaluated because its normalising constant $p_\psi(x)$ is intractable. We therefore introduce an inference network $q_\phi(\mathbf{z} \mid x)$, called the encoder, with parameters ϕ . It performs *amortised inference* by mapping each input to the parameters of its variational approximation.

We use a diagonal-Gaussian encoder,

$$q_\phi(\mathbf{z} \mid x) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_\phi(x), \text{diag}(\boldsymbol{\sigma}_\phi^2(x))), \quad (13.18)$$

where the encoder outputs the mean $\boldsymbol{\mu}_\phi(x)$ and positive standard deviation $\boldsymbol{\sigma}_\phi(x)$. For a single data point, the VAE objective is

$$\mathcal{L}(\psi, \phi; x) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z} \mid x)}[\log p_\psi(x \mid \mathbf{z})]}_{\text{Expected reconstruction log-likelihood}} - \underbrace{\text{D}_{\text{KL}}(q_\phi(\mathbf{z} \mid x) \parallel p(\mathbf{z}))}_{\text{Latent regulariser}}. \quad (13.19)$$

The total ELBO $\sum_{n=1}^N \mathcal{L}(\psi, \phi; x_n)$ is maximised jointly over ψ and ϕ by stochastic gradient ascent. The reconstruction term is estimated using reparameterised samples from $q_\phi(\mathbf{z} \mid x)$, while the KL term is often available in closed form, as it is for two Gaussian distributions. This joint training of a decoder and an amortised inference network is the defining construction of the variational autoencoder.

13.4 Limitations of Gaussian Mean-Field Variational Inference

Gaussian mean-field variational inference is a fast and scalable approximation that supports complex models such as BNNs and VAEs, but its simplicity rests on strong assumptions:

1. **Mean-field assumption:** the factorised approximation cannot represent posterior correlations between parameters and therefore loses the posterior’s covariance structure.
2. **Reverse-KL behaviour:** minimising $\text{D}_{\text{KL}}(q(\theta; \lambda) \parallel p(\theta \mid \mathcal{D}))$ strongly penalises an approximation that assigns non-negligible probability to regions where the posterior density is low. This mode-seeking behaviour can make q concentrate too narrowly and often contributes to underestimated marginal posterior variances.
3. **Gaussianity assumption:** a single Gaussian cannot represent skewness, heavy tails, or multimodality, and reverse-KL optimisation will often select one mode of a multimodal posterior.

These limitations motivate more expressive variational families, including structured approximations that relax the mean-field factorisation and function-space methods that define the approximation over induced distributions on functions.

Chapter 14

Dropout-based variational inference

So far, we have seen that performing variational inference requires optimising an approximating variational distribution. Interestingly, in the search for a suitable approximation, we can leverage a concept originally introduced for model regularisation: stochastic regularisation. This strategy relies on a simple intuition: injecting stochastic perturbations into the inference process at training time prevents any subset of parameters from co-adapting, forcing the network to distribute the learned representation across redundant pathways. Stochastic regularisation thus effectively acts as an implicit sampling mechanism: each perturbed configuration is nothing but a draw from an underlying distribution over network configurations. In this chapter, we will indeed discover that the intersection of stochastic regularisation techniques and Monte Carlo (MC) sampling can, under certain conditions, lead to an effective approximate variational distribution.

14.1 Monte Carlo dropout

As in all the VI strategies discussed in this book, we need to find a suitable Bayesian approximation to minimise the KL-divergence and deliver all the nice properties discussed in the introduction (Chapter 12). Since we adopt a Bayesian approach within the Deep Learning context (and thus, we deal with neural networks in many different ways), we should seek a strategy to treat the network’s parameters probabilistically. But wait, the literature on neural networks already explored randomisation inside the model: can’t we just exploit those strategies to lay the foundation for our Bayesian approximation? Of course, we can, and the method we will explore in this chapter is, indeed, based upon well-known pieces of math that have been around for many years: **stochastic regularisation techniques (SRTs)** [565]. As the name may suggest, SRTs serve as a regularisation mechanism for the model, achieved by introducing stochasticity in the predictive process. We can directly use this randomisation approach to derive the approximate distribution we seek throughout this chapter. We now discuss how it is possible to obtain a Bayesian approximation using one of the most renowned SRTs: **dropout** [1552]¹.

14.1.1 The dropout objective

Dropout is a powerful and popular SRT that is based on a simple, intuitive principle: randomly shutting neurons down during training. What does it mean, and why should we do such a thing? When we say shutting down, we literally mean *nullifying* the contribution a given neuron delivers to the output computation. For the sake of simplicity, consider a single hidden layer NN, with weight matrix $\mathbf{W}_h \in \mathbb{R}^{D \times Q}$ and biases $\mathbf{b}_h \in \mathbb{R}^Q$ (connecting the input to the *hidden* layer) and weight matrix $\mathbf{W}_l \in \mathbb{R}^{Q \times C}$ (connecting the hidden layer to the *last*, output layer). Under regular conditions, a given neuron $z^{(i)}$ on the hidden layer should account for a quantity which equals a non-linearly activated linear combination of the input and the vector $\mathbf{W}_h^{(i)}$ plus a constant $b_h^{(i)}$, i.e., $z^{(i)} = \sigma(\mathbf{x}^\top \mathbf{W}_h^{(i)} + b_h^{(i)})$. If we apply dropout on the hidden layer, some of the layer’s neurons $z^{(i)}$ will preserve their original value $\sigma(\mathbf{x}^\top \mathbf{W}_h^{(i)} + b_h^{(i)})$, while the remaining ones will be set to zero. Which and how many of these neurons will be zeroed out depends upon a parameter known as the **dropout rate**, which is the probability that

¹At this point in the chapter, acute readers could ask why one should pick dropout instead of any other SRT, which would be a legitimate question! There is no particular reason for using dropout instead of, for example, dropconnect, and it is indeed possible to derive a Bayesian approximation from other SRTs. For historical reasons, we will only explore dropout, since the original technique, commonly known as “Monte Carlo Dropout”, was originally proposed for dropout [566]. We leave to the most curious among the readers the demonstration of a possible Bayesian approximation using different SRTs.

a given neuron will be dropped out. Formally, this is equivalent to updating each hidden neuron by multiplying $\sigma(\mathbf{x}^\top \mathbf{W}_h^{(i)} + \mathbf{b}_h^{(i)}) \cdot \mathbf{o}^{(i)}$ with a random variable $\mathbf{o}^{(i)} \sim \mathcal{B}(1 - \phi)$, so that the neuron is zeroed with probability ϕ (the dropout rate mentioned earlier). The same mechanism can be applied to the input layer, dropping individual input components. This procedure will affect the final output $\hat{\mathbf{y}}$ of our predictor, and results in:

$$\hat{\mathbf{y}} = (\sigma(\mathbf{W}_h^\top (\mathbf{x} \circ \mathbf{o}_h) + \mathbf{b}_h) \circ \mathbf{o}_l)^\top \cdot \mathbf{W}_l, \quad (14.1)$$

where $\mathbf{o}_h = [\mathbf{o}_h^{(i)}]_{i=1}^D$ and $\mathbf{o}_l = [\mathbf{o}_l^{(i)}]_{i=1}^Q$ are the vectors containing the realisations $\mathbf{o}_h^{(i)} \sim \mathcal{B}(1 - \phi)$ and $\mathbf{o}_l^{(i)} \sim \mathcal{B}(1 - \phi)$ for every input $\mathbf{x}^{(i)}$ and hidden neuron $z^{(i)}$, and $\circ$ represents the Hadamard product. When the hidden neurons pass through the last layer of our example model, the final output no longer contains all the original information of the hidden layer, but only the contribution delivered by the unaltered neurons.

Now that we know how dropout works, the main question is why we should use it: isn't it unfavourable to drop some information out during the forward passes? Not really, if we remember that dropout is a regulariser that works on a neural network at training time. During training, dropping out some neurons would have the effect of forcing the predictor not to rely entirely on a tiny subset of neurons, because they can eventually not always be available. This behaviour avoids sparsity and encourages a dense knowledge distribution across the layer. During the learning process, the loss formulation will not be different from the usual one, with the exception that the output $\hat{\mathbf{y}}$ has been derived using dropout. As an example, in the regression case, we can use the squared error loss:

$$\mathcal{L}_{\text{dr}}(\mathcal{D}, \theta) := \frac{1}{2N} \sum_{n=1}^{N=|\mathcal{D}|} \|\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\|_2^2, \quad (14.2)$$

where θ denotes the model's parameters (i.e., $\mathbf{W}_h$, $\mathbf{b}_h$ and $\mathbf{W}_l$), used to compute the prediction $\hat{\mathbf{y}}^{(n)}$.

Starting from this dropout objective, we will now show how it is possible to derive its Bayesian approximation². Nevertheless, before diving into the details, we need to add a final ingredient to our objective to balance the accounts, that is, the L_2 regularisation on the network's weights:

$$\mathcal{L}_{\text{dr}}(\mathcal{D}, \theta) := \underbrace{\frac{1}{2N} \sum_{n=1}^{N=|\mathcal{D}|} \|\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\|_2^2}_{\text{Mean Squared Error}} + \underbrace{\lambda_{\mathbf{W}_h} \|\mathbf{W}_h\|_2^2 + \lambda_{\mathbf{W}_l} \|\mathbf{W}_l\|_2^2 + \lambda_{\mathbf{b}_h} \|\mathbf{b}_h\|_2^2}_{L_2}, \quad (14.3)$$

where $\lambda_{\mathbf{W}_h}$, $\lambda_{\mathbf{W}_l}$ and $\lambda_{\mathbf{b}_h}$ are the weight decay coefficients.

As we will discuss later in the chapter, including this term is a fundamental step for obtaining a principled approximation; luckily, this does not represent a practical obstacle since introducing this type of regularisation is a common practice in neural network optimisation.

In what follows, dropout is presented as a Bayesian approximation by following the original derivation of Gal and Ghahramani [566], which grounds the argument in Gaussian Processes. This is not the only possible route: an alternative derivation [565] dispenses with GPs altogether and works directly in weight space, casting dropout as variational inference in a Bayesian neural network with a mixture-of-Gaussians approximating distribution, and generalising the argument to the whole family of stochastic regularisation techniques. Although we follow the GP-based derivation, since it makes the equivalence with the dropout objective of Equation 14.2 explicit, the reader should bear in mind that the Bayesian interpretation of dropout does not necessarily depend on the GP machinery.

14.1.2 Gaussian Processes: the cornerstone of the approximation

In order to ascertain whether the dropout objective is a Bayesian approximation, we need to briefly anticipate a useful non-parametric model: the **Gaussian Processes** [1382]. For the sake of simplicity we will only cover the aspects strictly needed for our derivation; for a broader treatment of Gaussian Processes and kernels, see Chapter 23.

Gaussian Processes (GPs) are useful statistical tools that allow us to define distributions over functions. Let $f: \mathbb{R}^D \rightarrow \mathbb{R}$ be a *single-output* regressor;³ accordingly, we denote by $\mathbf{y} \in \mathbb{R}^N$ the vector of scalar targets and by $\mathbf{f} = [f(\mathbf{x}^{(n)})]_{n=1}^N \in \mathbb{R}^N$ the vector collecting the evaluations of f at the N inputs. So far, it is sufficient to assume

²Again, for the sake of simplicity, we will discuss the Bayesian approximation only for a specific case, i.e., the regression objective. The same properties apply for classification, even though some modifications are needed; we leave a discussion of this case to the reader as an exercise.

³The extension to the C outputs of our MLP is immediate, and is discussed at the end of the section.

the existence of a specific prior $p(\mathbf{f})$ over the elements of our hypotheses space, indicating which regressor $\mathbf{f}$ is more or less likely to have generated the data. Following from Bayes rule, we can obtain the posterior distribution as:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})} = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f}} \propto p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X}). \quad (14.4)$$

It is precisely now that GPs come into play, allowing us to compute the posterior analytically. For doing so, GPs exploit a clever intuition: similar inputs $\mathbf{x}$'s should result in similar outputs $\mathbf{f}(\mathbf{x})$'s; in a sense, we would like a model that makes the input $\mathbf{x}$'s co-vary. This behaviour can be implemented by employing a covariance function $k(\mathbf{x}_a, \mathbf{x}_b)$, which defines the similarity between two given inputs. The covariance function allows us to place a Gaussian distribution over the function values:

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(0, K(\mathbf{X}, \mathbf{X})) \quad (14.5)$$

$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \tau^{-1}\mathbf{I}_N), \quad (14.6)$$

where $K(\mathbf{X}, \mathbf{X})$ is an $N \times N$ covariance matrix defined using the covariance function $k(\mathbf{x}_a, \mathbf{x}_b)$ and $\tau^{-1}\mathbf{I}_N$ is the inherent predictive noise of the function, obtained by multiplying the inverse of the precision parameter τ with an $N \times N$ identity matrix $\mathbf{I}_N$. Note that the GP prior is a defined over the function $\mathbf{f}$, whereas Equation 14.5 states the distribution it induces over the finite set of evaluations $\mathbf{f}$ at the observed inputs; this is precisely what makes the model tractable.

The extension to the C -dimensional outputs of our MLP is immediate: it is sufficient to model each output component with its own regressor $\mathbf{f}_c : \mathbb{R}^D \rightarrow \mathbb{R}$, $c = 1, \dots, C$, placing an independent GP on each of them, all sharing the same covariance function. To keep the notation compact, we collect targets and function evaluations into two $N \times C$ matrices

$$\mathbf{Y} = [\mathbf{y}_1 \ \dots \ \mathbf{y}_C] \in \mathbb{R}^{N \times C}, \quad \mathbf{F} = [\mathbf{f}_1 \ \dots \ \mathbf{f}_C] \in \mathbb{R}^{N \times C}, \quad (14.7)$$

whose c -th columns $\mathbf{y}_c, \mathbf{f}_c \in \mathbb{R}^N$ gather, respectively, the c -th target component and the c -th function evaluations $[\mathbf{f}_c(\mathbf{x}^{(n)})]_{n=1}^N$ across the N samples; the n -th rows, which we denote by $\mathbf{y}^{(n)}, \mathbf{f}^{(n)} \in \mathbb{R}^C$, gather instead the C output components of the n -th sample. With this notation, Equation 14.5 holds column-wise,

$$\mathbf{f}_c|\mathbf{X} \sim \mathcal{N}(0, K(\mathbf{X}, \mathbf{X})) \quad (14.8)$$

$$\mathbf{y}_c|\mathbf{f}_c \sim \mathcal{N}(\mathbf{f}_c, \tau^{-1}\mathbf{I}_N), \quad (14.9)$$

with the $\mathbf{f}_c$'s mutually independent given the covariance function. In the following we write this set of C independent models compactly as $\mathbf{F}|\mathbf{X} \sim \mathcal{N}(0, K(\mathbf{X}, \mathbf{X}))$ and $\mathbf{Y}|\mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N)$, where the Gaussian is understood to act on each column separately. We do not need further details on GPs for our purpose.

At this point, a question remains unanswered: why do we need to introduce GPs at all? The reason is that GPs play a pivotal role in deriving our Bayesian approximation. Recall the ELBO derivation of Chapter 12, maximising the ELBO is equivalent to minimising the KL divergence between the real and the approximated distribution. For the dropout Bayesian approximation, we need to prove that the dropout objective is, in some sense, maximising the evidence lower bound. Starting from a GP on which we can derive the ELBO and using an appropriate covariance function, we can ultimately show that the ELBO objective is, indeed, equivalent to the dropout objective of Equation 14.2.

14.1.3 Dropout as a Bayesian approximation

Now that we have all the ingredients, we can discuss the details of the chapter's core: the use of dropout as a Bayesian approximation. The idea consists of three steps:

1. Modeling the structure of the MLP introduced in Section 14.1.1 using a Gaussian Process with convenient priors and covariance function;
2. Defining a suitable approximate distribution which simulates the effect of the dropout;
3. Finally, showing how maximising the ELBO leads to a natural equivalence with the dropout objective described in Equation 14.2.

Gaussian Process approximation of the MLP

We start by creating a Gaussian Process with a convenient covariance function; in particular, we would like to construct our GP in such a way that its approximation will be equivalent to optimising an MLP with dropout objective. Let's consider the following covariance function:

$$k_{p(\mathbf{w}_h), p(\mathbf{b}_h)}(\mathbf{x}_a, \mathbf{x}_b) = \int \int \sigma(\mathbf{w}_h^\top \mathbf{x}_a + \mathbf{b}_h) \sigma(\mathbf{w}_h^\top \mathbf{x}_b + \mathbf{b}_h) p(\mathbf{w}_h) p(\mathbf{b}_h) d\mathbf{w}_h d\mathbf{b}_h, \quad (14.10)$$

where, for convenience, we use the symbols $\mathbf{w}_h \in \mathbb{R}^D$ and $\mathbf{b}_h \in \mathbb{R}$ to represent a generic weight vector and bias term.⁴ Here we are denoting a specific covariance function (thus, a function which takes two inputs $\mathbf{x}_a$ and $\mathbf{x}_b$) but which also depends upon some probability distributions $p(\mathbf{w}_h)$ and $p(\mathbf{b}_h)$. Throughout this chapter we take standard normal priors, $p(\mathbf{w}_h) = \mathcal{N}(0, \mathbf{I}_D)$ and $p(\mathbf{b}_h) = \mathcal{N}(0, 1)$; this choice will be needed later, when evaluating the KL divergence in closed form. By sampling from these distributions we can consider a MC approximation of the integral, which can shift the dependence upon the distribution toward the dependence upon Q realisations drawn from these distribution. If we extract Q samples $\mathbf{W}_h^{(q)} \sim p(\mathbf{w}_h)$ and $\mathbf{b}_h^{(q)} \sim p(\mathbf{b}_h)$ from the prior probabilities, and denote $\mathbf{W}_h = [\mathbf{W}_h^{(q)}]_{q=1}^Q$ and $\mathbf{b}_h = [\mathbf{b}_h^{(q)}]_{q=1}^Q$, we can approximate Equation 14.10 as:

$$k_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{Q} \sum_{q=1}^Q \sigma(\mathbf{W}_h^{(q)\top} \mathbf{x}_a + \mathbf{b}_h^{(q)}) \sigma(\mathbf{W}_h^{(q)\top} \mathbf{x}_b + \mathbf{b}_h^{(q)}). \quad (14.11)$$

Then, we use $\mathbf{W}_h$ and $\mathbf{b}_h$ for deriving a graphical model analogously to Equation 14.5:

$$\mathbf{f}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h \sim \mathcal{N}(0, K_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{X}, \mathbf{X})) \quad (14.12)$$

$$\mathbf{y}_c | \mathbf{f}_c \sim \mathcal{N}(\mathbf{f}_c, \tau^{-1} \mathbf{I}_N), \quad (14.13)$$

where $K_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{X}, \mathbf{X})$, as in Equation 14.5, denotes an $N \times N$ covariance matrix derived from applying the function $k_{\mathbf{W}_h, \mathbf{b}_h}$ to each input pair in $(\mathbf{X}, \mathbf{X})$. Having placed an independent GP on each output component, conditioning on $\mathbf{W}_h$ and $\mathbf{b}_h$ makes the columns of $\mathbf{Y}$ conditionally independent, so that $p(\mathbf{Y} | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h)$ factorises over the components $\mathbf{y}_c$; this gives us the density of the whole matrix $\mathbf{Y}$, from which we can derive $p(\mathbf{Y} | \mathbf{X})$ from marginalising $\mathbf{W}_h$ and $\mathbf{b}_h$:

$$p(\mathbf{Y} | \mathbf{X}) = \int \int p(\mathbf{Y} | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h \quad (14.14)$$

$$= \int \int \prod_{c=1}^C p(\mathbf{y}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h. \quad (14.15)$$

Then, the density function $p(\mathbf{y}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h)$ can be obtained from marginalising $\mathbf{f}_c$, which allow us to expand $p(\mathbf{Y} | \mathbf{X})$ as:

$$p(\mathbf{Y} | \mathbf{X}) = \int \int \left(\int \prod_{c=1}^C p(\mathbf{y}_c | \mathbf{f}_c, \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) p(\mathbf{f}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) d\mathbf{f}_c \right) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h \quad (14.16)$$

$$= \int \int \left(\prod_{c=1}^C \int p(\mathbf{y}_c | \mathbf{f}_c) p(\mathbf{f}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) d\mathbf{f}_c \right) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h. \quad (14.17)$$

We can then compute each term $p(\mathbf{y}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) = \int p(\mathbf{y}_c | \mathbf{f}_c) p(\mathbf{f}_c | \mathbf{X}, \mathbf{W}_h, \mathbf{b}_h) d\mathbf{f}_c$ analytically using Equation 14.12, as $\mathcal{N}(0, K_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{X}, \mathbf{X}) + \tau^{-1} \mathbf{I}_N)$, and rewrite $p(\mathbf{Y} | \mathbf{X})$ as:

$$p(\mathbf{Y} | \mathbf{X}) = \int \int \prod_{c=1}^C \mathcal{N}(\mathbf{y}_c; 0, K_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{X}, \mathbf{X}) + \tau^{-1} \mathbf{I}_N) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h. \quad (14.18)$$

Our graphical model encodes now the parameters $\mathbf{W}_h$ and $\mathbf{b}_h$; to bring this formulation closer to the one used for a multi-layer perceptron, we need to introduce auxiliary variables. We can factorise the covariance matrix as

⁴Reading this chapter after Chapter 24 may rise the question: is Equation 14.10 a *valid* covariance function? The answer is yes, but for the sake of simplicity, we omit the proof. In this regard, we suggest to the more curious among the readers to delve deeper into the discussion around Marginalised Kernels [1639].

an outer product $\Psi\Psi^\top$, where $\Psi = [\psi(\mathbf{x}^{(n)}, \mathbf{W}_h, \mathbf{b}_h)^\top]_{n=1}^N \in \mathbb{R}^{N \times Q}$ is the matrix built from the Q -dimensional vectors $\psi(\mathbf{x}, \mathbf{W}_h, \mathbf{b}_h) = \sqrt{\frac{1}{Q}}\sigma(\mathbf{W}_h^\top \mathbf{x} + \mathbf{b}_h)$, so that $K_{\mathbf{W}_h, \mathbf{b}_h}(\mathbf{X}, \mathbf{X}) = \Psi\Psi^\top$. Replacing this in Equation 14.18 yields:

$$p(\mathbf{Y}|\mathbf{X}) = \int \int \prod_{c=1}^C \mathcal{N}(\mathbf{y}_c; 0, \Psi\Psi^\top + \tau^{-1}\mathbf{I}_N) p(\mathbf{W}_h) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{b}_h. \quad (14.19)$$

For each element on this product, we introduce a $Q \times 1$ variable $\mathbf{w}_c \sim \mathcal{N}(0, \mathbf{I}_Q)$:

$$\mathcal{N}(\mathbf{y}_c; 0, \Psi\Psi^\top + \tau^{-1}\mathbf{I}_N) = \int \mathcal{N}(\mathbf{y}_c; \Psi\mathbf{w}_c, \tau^{-1}\mathbf{I}_N) \mathcal{N}(\mathbf{w}_c; 0, \mathbf{I}_Q) d\mathbf{w}_c. \quad (14.20)$$

Note that now the mean of the independent $\mathbf{y}_c$'s has the form $\Psi\mathbf{w}_c$, with $\psi(\mathbf{x}^{(n)}, \mathbf{W}_h, \mathbf{b}_h)^\top \mathbf{w}_c$ for the n -th sample; by collecting the auxiliary variables in matrix notation as $\mathbf{W}_l = [\mathbf{w}_c]_{c=1}^C$ we recover the mean prediction $\mathbf{Y} = \Psi\mathbf{W}_l$, which is now more similar to the original MLP formulation. We can then write $p(\mathbf{Y}|\mathbf{X})$ as a marginalisation over $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$:

$$p(\mathbf{Y}|\mathbf{X}) = \int \int \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) p(\mathbf{W}_h) p(\mathbf{W}_l) p(\mathbf{b}_h) d\mathbf{W}_h d\mathbf{W}_l d\mathbf{b}_h. \quad (14.21)$$

Approximated distribution prior

Once having defined the GP we need to construct the approximated distribution for our model parameters. The idea should be to find a suitable distribution $q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)$ which can minimise the KL divergence with the true distribution $p(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)$. Since we assumed that the parameters $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$ are mutually independent we can derive the approximated distribution $q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)$ as a product of each individual approximated component:

$$q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) = q(\mathbf{W}_h) q(\mathbf{W}_l) q(\mathbf{b}_h). \quad (14.22)$$

Now, remember the main goal of this whole section: we are not seeking a generic approximation, but a *dropout*-based one. In a sense, we would like to model our independent $q(\mathbf{W}_h)$, $q(\mathbf{W}_l)$, and $q(\mathbf{b}_h)$ as the product of a stochastic process involving dropout. But how does the effect of dropout impacts the parameters' probability distribution? Dropping an input neuron $\mathbf{x}_d^{(n)}$ is equivalent to setting to zero all the weights connecting that neuron to the hidden layer. Hence, if we set the same dropout rate ϕ for all the neurons on a given layer, we approximate the individual $\mathbf{W}_h^{(i)}$ and $\mathbf{W}_l^{(i)}$ using a mixture of Gaussians with shared covariance and individual mean vectors, and a simple Gaussian for the biases $\mathbf{b}_h^{(i)}$:

$$q(\mathbf{W}_h^{(i)}) = (1 - \phi) \mathcal{N}(\mathbf{M}_h^{(i)}, \sigma^2 \mathbf{I}_Q) + \phi \mathcal{N}(0, \sigma^2 \mathbf{I}_Q), \quad (14.23)$$

$$q(\mathbf{W}_l^{(i)}) = (1 - \phi) \mathcal{N}(\mathbf{M}_l^{(i)}, \sigma^2 \mathbf{I}_C) + \phi \mathcal{N}(0, \sigma^2 \mathbf{I}_C), \quad (14.24)$$

$$q(\mathbf{b}_h) = \mathcal{N}(\mathbf{m}_h, \sigma^2 \mathbf{I}_Q). \quad (14.25)$$

The matrices $\mathbf{M}_h = [\mathbf{M}_h^{(i)}]_{i=1}^D$ and $\mathbf{M}_l = [\mathbf{M}_l^{(i)}]_{i=1}^Q$, in a sense, encode the network weights in absence of dropout: the probabilistic weights $\mathbf{W}_h$ and $\mathbf{W}_l$ are thus obtained by drawing from a mixture having $1 - \phi$ probability of returning the "actual weight", and ϕ probability of zeroing them out.

Starting from Equation 14.23 we can finally reparametrise the probabilities over $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$ with respect to Bernoulli random variables, precisely as it happens by using dropout. With the same approach of the reparametrisation trick used for AutoEncoders [892], we can reduce the stochasticity to simple events (noise generation and dropout) and compose the probability on the parameters directly with respect to them. First, we need to generate the Bernoulli random variables. We can generate a vector $\mathbf{o}_h \in \{0, 1\}^D$ where each element $\mathbf{o}_h^{(i)} \sim \mathcal{B}(1 - \phi)$ follows the Bernoulli distribution with probability $1 - \phi$; then, it is sufficient to multiply this random vector by the mean matrix $\text{diag}(\mathbf{o}_h) \cdot \mathbf{M}_h$ for zeroing the rows corresponding to the dropped units. An analogous approach can be implemented for modelling dropout on the hidden layer, by introducing a vector $\mathbf{o}_l \in \{0, 1\}^Q$. Secondly, we generate the noise in Equation 14.23 by introducing the support random variables $\mathbf{R}_h \sim \mathcal{N}(0, \mathbf{I}_{D \times Q})$, $\mathbf{R}_l \sim \mathcal{N}(0, \mathbf{I}_{Q \times C})$ and $\mathbf{r}_h \sim \mathcal{N}(0, \mathbf{I}_Q)$ for computing the additive noise. By combining these two steps we can derive $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$ as a function of $\mathbf{o}_h$, $\mathbf{o}_l$, $\mathbf{R}_h$, $\mathbf{R}_l$ and $\mathbf{r}_h$:

$$\mathbf{W}_h(\mathbf{o}_h, \mathbf{R}_h) = \text{diag}(\mathbf{o}_h) \cdot (\mathbf{M}_h + \sigma \mathbf{R}_h) + \text{diag}(1 - \mathbf{o}_h) \cdot \sigma \mathbf{R}_h \quad (14.26)$$

$$\mathbf{W}_l(\mathbf{o}_l, \mathbf{R}_l) = \text{diag}(\mathbf{o}_l) \cdot (\mathbf{M}_l + \sigma \mathbf{R}_l) + \text{diag}(1 - \mathbf{o}_l) \cdot \sigma \mathbf{R}_l \quad (14.27)$$

$$\mathbf{b}_h(\mathbf{r}_h) = \mathbf{m}_h + \sigma \mathbf{r}_h. \quad (14.28)$$

Note that the sampled weights still carry the Gaussian perturbation $\sigma \mathbf{R}_h$ and $\sigma \mathbf{R}_l$: the equivalence with dropout is recovered in the limit $\sigma \rightarrow 0$, for which $\mathbf{W}_h(\mathbf{o}_h, \mathbf{R}_h) \rightarrow \text{diag}(\mathbf{o}_h) \cdot \mathbf{M}_h$ and $\mathbf{W}_l(\mathbf{o}_l, \mathbf{R}_l) \rightarrow \text{diag}(\mathbf{o}_l) \cdot \mathbf{M}_l$, i.e. exactly the masked weights of Section 14.1.1.

We can finally exploit these new reparametrisations for evaluating the evidence lower bound.

Evaluating the ELBO

The notation introduced in Equation 14.26 – thus, separating the random variables from the actual network parameters – is fundamental for facilitating the computation of the ELBO. Recalling again the ELBO derivation of Chapter 12 we obtain:

$$\mathcal{L}_{GP-VI} := \int \int \int q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) d\mathbf{W}_h d\mathbf{W}_l d\mathbf{b}_h - \text{KL}(q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) || p(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)). \quad (14.29)$$

Let us begin by computing the $\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)$ term; for doing so, we can exploit the independence between the $\mathbf{y}_c$'s:

$$\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) = \log \prod_{c=1}^C p(\mathbf{y}_c|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \quad (14.30)$$

$$= \sum_{c=1}^C \log p(\mathbf{y}_c|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \quad (14.31)$$

$$= \sum_{c=1}^C \log \mathcal{N}(\mathbf{y}_c; \Psi \mathbf{W}_l^{(c)}, \tau^{-1} \mathbf{I}_N) \quad (14.32)$$

$$= -\frac{NC}{2} \log(2\pi) + \frac{NC}{2} \log(\tau) - \sum_{c=1}^C \frac{\tau}{2} \|\mathbf{y}_c - \Psi \mathbf{W}_l^{(c)}\|_2^2. \quad (14.33)$$

Now, let $\hat{\mathbf{Y}} = \Psi \mathbf{W}_l$, with $\hat{\mathbf{y}}^{(n)} = \psi(\mathbf{x}^{(n)}, \mathbf{W}_h, \mathbf{b}_h)^\top \mathbf{W}_l$: by doing so, we obtain a prediction matrix $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times C}$, and can thus sum over the first dimension of this matrix:

$$\sum_{c=1}^C \|\mathbf{y}_c - \hat{\mathbf{y}}_c\|_2^2 = \sum_{n=1}^N \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|_2^2. \quad (14.34)$$

We can thus rewrite the first term in the integral of the $\mathcal{L}_{GP-VI}$ objective as a sum over the samples:

$$q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) = \quad (14.35)$$

$$= \sum_{n=1}^N q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) \quad (14.36)$$

$$= \sum_{n=1}^N q(\mathbf{o}_h, \mathbf{R}_h, \mathbf{o}_l, \mathbf{R}_l, \mathbf{r}_h) \log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{W}_h(\mathbf{o}_h, \mathbf{R}_h), \mathbf{W}_l(\mathbf{o}_l, \mathbf{R}_l), \mathbf{b}_h(\mathbf{r}_h)). \quad (14.37)$$

Now, the integral in Equation 14.29 can be approximated using MC integration. Specifically, instead of integrating over $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$ we now sum over a set of realisations $\tilde{\mathbf{W}}_h$, $\tilde{\mathbf{W}}_l$ and $\tilde{\mathbf{b}}_h$.

$$\mathcal{L}_{GP-MC} := \sum_{n=1}^N \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \tilde{\mathbf{W}}_h^{(n)}, \tilde{\mathbf{W}}_l^{(n)}, \tilde{\mathbf{b}}_h^{(n)}) - \text{KL}(q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) || p(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)). \quad (14.38)$$

Lastly, we need to compute the KL divergence between the real and the approximated distribution. The main obstacle here is represented by the absence of an analytic solution that solves our problem, since we would compare a mixture of Gaussians against a single Gaussian distribution. Since both q and p factorise over $\mathbf{W}_h$, $\mathbf{W}_l$ and $\mathbf{b}_h$, however, the divergence splits *exactly* into the sum of three independent terms:

$$\begin{aligned} & \text{KL}(q(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h) || p(\mathbf{W}_h, \mathbf{W}_l, \mathbf{b}_h)) \\ &= \text{KL}(q(\mathbf{W}_h) || p(\mathbf{W}_h)) + \text{KL}(q(\mathbf{W}_l) || p(\mathbf{W}_l)) + \text{KL}(q(\mathbf{b}_h) || p(\mathbf{b}_h)), \end{aligned} \quad (14.39)$$

so that the only approximation needed concerns the two mixture terms⁵. For a large enough Q we can approximate the KL on $\mathbf{W}_h$ and $\mathbf{W}_l$ as:

$$\text{KL}(q(\mathbf{W}_h)||p(\mathbf{W}_h)) \approx \frac{\text{DQ}}{2}(\sigma^2 - \log(\sigma^2) - 1) + \frac{1-\phi}{2} \sum_{d=1}^D \mathbf{M}_h^{(d)\top} \mathbf{M}_h^{(d)} + \text{const} \quad (14.40)$$

$$\text{KL}(q(\mathbf{W}_l)||p(\mathbf{W}_l)) \approx \frac{\text{QC}}{2}(\sigma^2 - \log(\sigma^2) - 1) + \frac{1-\phi}{2} \sum_{q=1}^Q \mathbf{M}_l^{(q)\top} \mathbf{M}_l^{(q)} + \text{const.} \quad (14.41)$$

The bias term can be evaluated analytically as:

$$\text{KL}(q(\mathbf{b}_h)||p(\mathbf{b}_h)) = \frac{1}{2}(\mathbf{m}_h^\top \mathbf{m}_h + Q(\sigma^2 - \log(\sigma^2) - 1)) + \text{const} \quad (14.42)$$

By combining the individual KL terms for the weights and biases terms, ignoring σ (which do not affect the optimisation over $\mathbf{M}_h$, $\mathbf{M}_l$, and $\mathbf{m}_h$) and rescaling by $\frac{1}{N\tau}$, we finally obtain the MC Bayesian approximation as:

$$\mathcal{L}_{GP-MC} \propto -\frac{\tau}{2} \sum_{n=1}^N \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|_2^2 - \frac{1-\phi}{2} (\|\mathbf{M}_h\|_2^2 + \|\mathbf{M}_l\|_2^2) - \frac{1}{2} \|\mathbf{m}_h\|_2^2 \quad (14.43)$$

$$\propto -\frac{1}{2N} \sum_{n=1}^N \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|_2^2 - \frac{1-\phi}{2N\tau} (\|\mathbf{M}_h\|_2^2 + \|\mathbf{M}_l\|_2^2) - \frac{1}{2N\tau} \|\mathbf{m}_h\|_2^2. \quad (14.44)$$

Since maximising $\mathcal{L}_{GP-MC}$ is equivalent to minimising $-\mathcal{L}_{GP-MC}$, the expression above recovers the dropout objective in Equation 14.3 for the choice $\lambda_{\mathbf{W}_h} = \lambda_{\mathbf{W}_l} = \frac{1-\phi}{2N\tau}$ and $\lambda_{\mathbf{b}_h} = \frac{1}{2N\tau}$. Note that this correspondence fixes the model precision τ once the weight decay has been chosen. This shows that using a dropout objective along with a proper regularisation is, indeed, a proper Bayesian approximation.

14.2 Concrete Dropout: dropout rate as a variational parameter

The derivation above tells us that dropout *is* a Bayesian approximation, but it says nothing about how the “optimal” value of the dropout rate ϕ should be chosen. A related question, for a neural network with several layers, is whether *different* rates should be used across layers (or even, at the extreme, across individual neurons). An apparently reasonable approach is a grid-search procedure over a given set of candidate dropout rates, to estimate the best value for each layer. This procedure is however an unfeasible solution: it is prohibitively expensive for large models, also taking into account that the number of per-layer dropout configurations grows exponentially with the number of layers (let alone the combinatorial blow-up of a per-neuron configuration!). Moreover, it becomes altogether impractical in settings where the amount of available data changes over time, such as in a reinforcement learning or lifelong learning, since the optimal rate would have to be re-tuned as training progresses.

An alternative and more feasible approach is to *learn* the dropout rate itself, alongside the network weights, using gradient descent, by minimising a suitable objective function. To this aim, the variational interpretation of dropout, which takes the name of **Concrete Dropout** [568], provides an elegant solution: since the dropout rate parametrises the approximating distribution of Equation 14.23, it is a variational parameter on exactly the same footing as $\mathbf{M}_h$, $\mathbf{M}_l$ and $\mathbf{m}_h$, and the ELBO itself is the objective to be optimised with respect to it. Writing it for a generic L -layer network with per-layer rates ϕ_l and mean weight matrices $\mathbf{M}_l$, and normalising by N , we obtain

$$\hat{\mathcal{L}} = -\frac{1}{N} \sum_{n=1}^N \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \hat{\omega}_n) + \frac{1}{N} \sum_{l=1}^L \text{KL}(q_{\phi_l}(\mathbf{W}_l) || p(\mathbf{W}_l)) , \quad (14.45)$$

where $\hat{\omega}_n \sim q$ denotes a single set of weights sampled through the dropout masks, and the per-layer divergence takes the form

$$\text{KL}(q_{\phi_l}(\mathbf{W}_l) || p(\mathbf{W}_l)) \propto \frac{1-\phi_l}{2} \|\mathbf{M}_l\|_2^2 - K_l \mathcal{H}(\phi_l) , \quad (14.46)$$

with K_l the number of units of layer l and

$$\mathcal{H}(\phi_l) = -\phi_l \log \phi_l - (1 - \phi_l) \log(1 - \phi_l) \quad (14.47)$$

⁵At this stage, one could legitimately ask: yet another approximation? Unfortunately, yes. Since going into this level of detail would weigh the argument down, we do not discuss these approximations here. For a more comprehensive treatment, please refer to [566].

the entropy of a Bernoulli random variable of rate ϕ_l . The dropout rate therefore ceases to be a hyperparameter to be searched for, and becomes the outcome of the optimisation itself. This, however, is only possible if the gradient of Equation 14.45 with respect to ϕ_l can actually be computed, which is not the case for the discrete masks used so far.

The key idea for overcoming this issue is to reparametrise the dropout mask itself, so that one can legitimately backpropagate through the “randomised” neurons (that is, through the very units switched off by the dropout layer). Such reparametrisation tricks, which also lay the groundwork for techniques like Variational Autoencoders (VAEs) [892], will be treated in detail in later chapters; here we limit ourselves to the Gumbel-softmax trick [831], which is precisely what allows the dropout rate to be treated as an actual variational parameter during optimisation. Since in the case of dropout we have to optimise the parameters of a *discrete* Bernoulli distribution, the original reparametrisation trick cannot be applied directly: sampling a binary mask is not differentiable with respect to the dropout rate ϕ . A continuous relaxation is therefore required, such as the Concrete distribution [1115], originally proposed for latent random variables in deep generative models. The Concrete distribution replaces the $\arg \max$ appearing in the Gumbel-max reparametrisation of a discrete variable with a temperature-controlled softmax, thus yielding a continuous, differentiable random variable which concentrates on the vertices of the simplex as the temperature $t \rightarrow 0$. This allows a discrete, and thus non-differentiable, random variable $\mathbf{z}$ to be rewritten as $\tilde{\mathbf{z}} = g(\phi, \varepsilon)$ where the whole stochasticity is carried by a random variable ε which does not depend on ϕ , and g is deterministic and differentiable in ϕ . Such a factorisation is precisely what is needed in our case: the dropout masks make the units of the network random variables, and applying the above trick to them allows the gradient to be back-propagated onto the variational parameter ϕ that we are trying to optimise. In the binary case of interest here, the softmax reduces to the sigmoid $\sigma(\cdot)$, and the relaxed mask reads

$$\tilde{\mathbf{z}} = \sigma \left(\frac{1}{t} (\log \phi - \log(1 - \phi) + \log u - \log(1 - u)) \right), \quad (14.48)$$

where $u \sim \mathcal{U}[0, 1]$ is a uniformly distributed variable which does not depend on the parameter ϕ . Gradients therefore flow only through the terms involving ϕ , whereas u acts as an external source of noise, making now $\tilde{\mathbf{z}}$ differentiable with respect to ϕ .

14.3 Dropout injection

The original MC Dropout formulation envisages activating dropout at inference time on a neural network which uses dropout for regularisation during training. If, on the one hand, this technique yields a solution with minimal implementation overhead, on the other hand, its integration within already trained neural networks would require retraining the model. This is often infeasible, either because the original training pipeline or data is no longer available, or because retraining risks degrading the performance of a deployed model that has already undergone a validation process. As a solution to this issue, recent work shows that such a strategy can be applied to arbitrary NNs, regardless of whether or not dropout was used during training; this technique is known as **Dropout Injection** [996]. Such a strategy avoids the costs and pitfalls of retraining altogether, retrofitting uncertainty estimation onto an existing deterministic network without altering its weights or requiring further optimisation. The first solution proposed in the literature stems from a simple idea, i.e., finding a suitable dropout rate ϕ *after training* by minimising the Negative Log-Likelihood (NLL) [1076]. Albeit intuitive in practice, injecting dropout into an already trained neural network hides subtle implications which have not been addressed in the original formulation. Specifically, a noteworthy drawback comes to light when using this strategy for regression problems.

14.3.1 Dropout injection: the regression case

In a regression problem, the predictor $f(\mathbf{x}, \theta)$ with parameters θ takes as input a D -dimensional $\mathbf{x}$ vector and outputs a real number $\mathbf{y}$. Consider now a Monte Carlo Variational Inference (MC-VI) regressor $f_{\text{MC-VI}} : \mathbb{R}^D \rightarrow \mathbb{R}^M$ which takes as input a D -dimensional vector and outputs, instead, M MC predictions by sampling from a parameters’ distribution $q(\theta|\phi)$ approximated using a dropout rate ϕ . Following Loquercio et al., the main challenge of dropout injection would consist in finding the most suitable dropout rate ϕ which minimises the NLL:

$$\phi = \arg \min_{\phi'} \frac{1}{N} \sum_{n=1}^N \frac{1}{2} \frac{(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi'))^2}{\sigma^{2(n)}(\phi')} + \frac{1}{2} \log(\sigma^{2(n)}(\phi')), \quad (14.49)$$

where $\hat{\mathbf{y}}^{(n)}(\phi')$ and $\sigma^{2(n)}(\phi')$ are the empirical mean and variance of the prediction computed using the MC-VI regressor; it is worth noting that both of them depend upon the dropout rate ϕ' . Practically, too high ϕ' values

will, at some point, lead to a severe degradation of the performances, since the Bayesian model average (BMA) $\frac{1}{M} \sum_{m=1}^M f(\mathbf{x}, \theta_m)$ would deviate too much from the prediction of the original, deterministic network $f(\mathbf{x}, \theta_{\text{MAP}})$. Hence, the range of suitable ϕ values shrinks to the interval $[0, \phi_{\text{max}}]$, where ϕ_{max} represents the dropout rate for which the performance degradation is tolerable.

In this situation, however, a subtle complication may eventually occur. In principle, one can minimise the NLL either by minimising the squared error or by making the uncertainty measure equal to the squared error for each sample. Hence, the problem arises when for the maximum value ϕ_{max} the variances captured by $\sigma^{2(n)}(\phi)$ are too small to accurately reflect the squared errors $(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi))^2$. In the context of dropout injection, where it is not possible to tune the predictions, the NLL minimisation serves only to identify the value of ϕ which minimises the distance between the squared error and the empirical variance $|\sigma^{2(n)}(\phi) - (\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi))^2|$. But what happens if the range $[0, \phi_{\text{max}}]$ is so small that it will always produce variances significantly lower than the corresponding squared errors? In such a situation, even though the uncertainty measure cannot accurately reflect the errors, we can approach the problem from a different angle: if the difficulties arise from the insufficient magnitude of the uncertainty measure, its rescaling provides a practical solution. By doing so, we can rewrite Equation 14.49 as a two-variable minimisation problem, where we search for the optimal ϕ and scale factor S jointly:

$$\phi, S = \arg \min_{\phi', s} \frac{1}{N} \sum_{n=1}^N \frac{1}{2} \frac{(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi'))^2}{s \cdot \sigma^{2(n)}(\phi')} + \frac{1}{2} \log(s \cdot \sigma^{2(n)}(\phi')), \quad (14.50)$$

Notably, one can easily demonstrate that it is possible to compute the optimal scale factor S over a set of data, given ϕ , in closed form. Specifically and originally proposed for calibrating regressors, the solution to this problem takes the name of σ -scaling [981]. The optimal scale factor $s(\phi) = \frac{1}{N} \sum_{n=1}^N [(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi))^2 \cdot \sigma_n^{-2}(\phi)]$ can be computed as the expectation of the squared error divided by the variance. The proof is left to the reader as an exercise. By doing so, the problem of Equation 14.50 can be formulated as a simpler optimisation problem:

$$\phi = \arg \min_{\phi'} \frac{1}{N} \sum_{n=1}^N \frac{1}{2} \frac{(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}(\phi'))^2}{\sigma^{2(n)}(\phi') \cdot s(\phi')} + \frac{1}{2} \log(\sigma^{2(n)}(\phi') \cdot s(\phi')). \quad (14.51)$$

Optimising in this way means the search is no longer confined to a fixed one-dimensional curve, but ranges freely across the two-dimensional surface in which that optimal scale curve $s(\phi)$ is embedded, as illustrated in Figure 14.1. In the scale-agnostic case the search is confined to the slice $s = 1$, and the star marks the best (sub-optimal) NLL attainable there; rescaling the uncertainty measure instead grants access to regions of the surface that would otherwise remain unreachable, yielding a different dropout rate with a lower NLL.

14.3.2 Dropout Injection for classification

When discussing Dropout Injection for regression problems, we revealed the existence of an issue related to the scale of the approximated distribution. Interestingly, the same does not hold for classification problems, as it becomes evident when modelling the same conditions for classifiers. For regression problems, the function $f(\mathbf{x}, \theta)$ outputs a scalar $\hat{\mathbf{y}} \in \mathbb{R}$: Here, the output can be modelled by placing a Gaussian on the predictive distribution $f(x, \theta) \sim \mathcal{N}(\mu, \sigma^2)$. For the classification case, instead, the function $f(\mathbf{x}, \theta)$ returns a probability vector $p = \hat{\mathbf{y}} = \text{softmax}(z)$, with $z \in \mathbb{R}^C$: in this case, since $\hat{\mathbf{y}}$ represent a vector on the C -dimensional simplex, the Dirichlet distribution is a convenient choice for modelling the posterior $\hat{\mathbf{y}} \sim \text{Dir}(\alpha)$.

The key architectural difference between classification and regression models is the use of a softmax activation in the final layer for classification. This enables moment matching to be carried out directly on the logits z , which can then be used to estimate the parameters of the Dirichlet distribution. The main challenge in this approach lies in identifying an appropriate distribution over the logits, such that the softmax-transformed outputs follow a Dirichlet distribution. Thankfully, known results on Bayesian statistics tell us that the softmax of independent gamma variables follows a Dirichlet distribution [420]. We can exploit this result by modelling the exponential of each logit using independent Gamma distributions:

$$e^{z_i} \sim \Gamma(\alpha_i, \frac{1}{\lambda_i}), \forall i \in [1, C] \implies \text{softmax}(z) \sim \text{Dir}(\alpha). \quad (14.52)$$

We can now perform the moment matching operation directly on the independent Gamma distributions, estimating, for each of them, an empirical mean $\hat{\mu}_i$ and variance $\hat{\sigma}_i^2$ for deriving the shape α_i and scale $\frac{1}{\lambda_i}$ parameters:

$$\begin{bmatrix} \alpha_i \lambda_i^{-1} \\ \alpha_i \lambda_i^{-2} \end{bmatrix} = \begin{bmatrix} \hat{\mu}_i \\ \hat{\sigma}_i^2 \end{bmatrix} \implies \alpha_i = \hat{\mu}_i^2 \hat{\sigma}_i^{-2}, \frac{1}{\lambda_i} = \hat{\sigma}_i^2 \hat{\mu}_i^{-1} \quad (14.53)$$

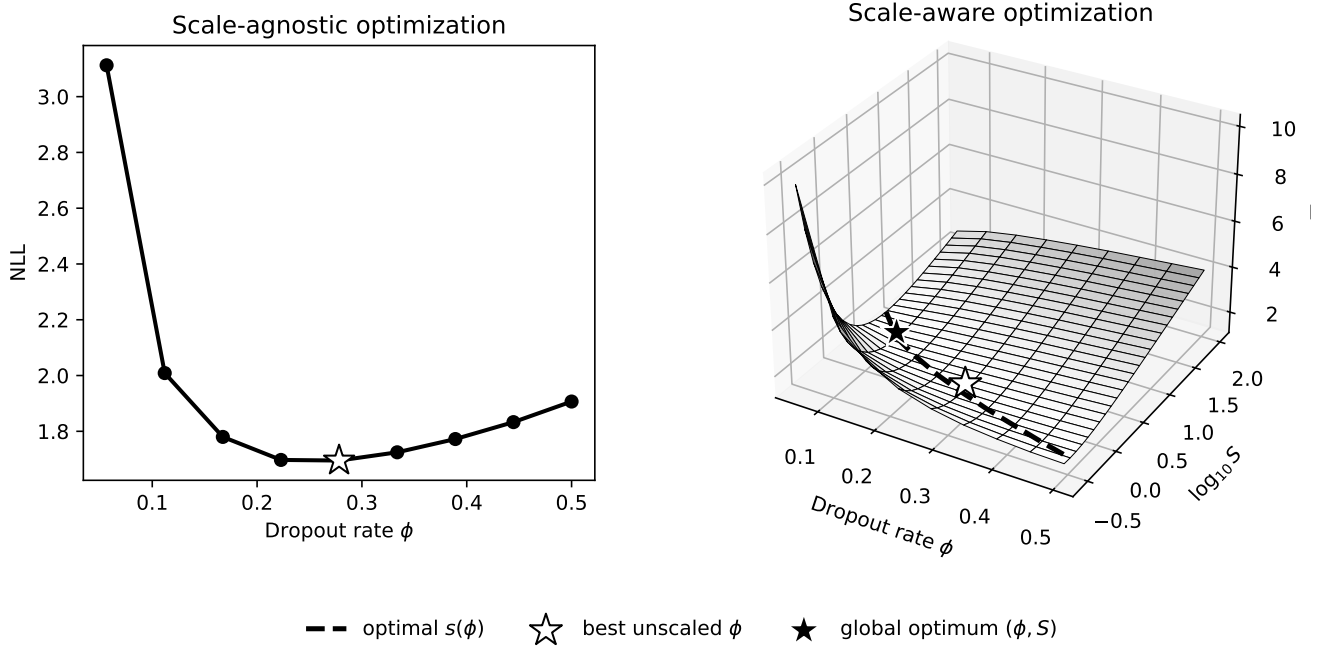


Figure 14.1: Comparison between the scale-agnostic and the scale-aware formulations of the injected-dropout optimisation problem, for an MLP on the *Boston Housing* data set. **Left:** the NLL of Equation 14.49 as a function of the dropout rate ϕ alone. **Right:** the NLL surface of Equation 14.50, considered as a function of both ϕ and the scale factor s ; the dashed curve traces the optimal scale factor $s(\phi)$ of Equation 14.51, the white star reproduces the scale-agnostic solution and the black one the global optimum (ϕ, S) .

We can then derive the parameters of the resulting Dirichlet on the probability vectors:

$$p \sim \text{Dir}(\hat{\mu}^2 \circ \hat{\sigma}^{-2}). \quad (14.54)$$

The consequence of scaling the empirical variance in this setup is straightforward. The exponential logits would now follow a Gamma distribution with scaled shape and rate parameters, changing the Dirichlet parameters on the output probability vector accordingly:

$$\Gamma(\hat{\mu}_i^2(s^{-1} \cdot \hat{\sigma}_i^{-2}), s \cdot \hat{\sigma}_i^2 \hat{\mu}_i^{-1}) \implies \text{Dir}(\hat{\mu}^2 \circ (s^{-1} \cdot \hat{\sigma}^{-2})). \quad (14.55)$$

Since the Dirichlet mean is preserved when scaling, so does the NLL. Hence, when dealing with classification, scaling the empirical variance on the logits do not affect the NLL.

14.4 Concluding remarks

In this chapter we saw that a technique devised for regularisation can be reinterpreted, under the right assumptions, as approximate Bayesian inference. It is precisely this simplicity that makes the method usable at scale: indeed, dropout-based inference does not require any architectural or optimisation change, and the entire cost of estimating the predictive uncertainty is deferred to the inference phase. On top of this, the possibility of applying the technique to networks that have *already* been trained opens the door to a wide range of practical scenarios, making dropout-based inference a quick solution for experts and practitioners alike. Nevertheless we have to keep in mind that the multiple forward passes required at inference time represent a non-negligible obstacle, particularly as model size increases, where the computation of a single prediction can no longer be taken for granted.

To close the chapter, it is worth drawing attention to a distinctive feature of dropout-based inference: unlike more explicit variational treatments, it introduces no variational parameters beyond those a deterministic network already possesses (with the exception of Concrete Dropout), and attaches no distribution to individual weights or neurons. From the next chapter until the end of this part, we shall see that this is the exception rather than the rule: the methods that follow make the variational object explicit – defined over weights, over functions, or through a set of particles – and consequently require additional quantities to be specified, optimised, or propagated.

Chapter 15

Natural-gradient-based variational inference

Variational inference can make the optimization of large Bayesian neural networks tractable by restricting optimization to a specific class of distributions. Here, we discuss natural-gradient-based methods, where gradients are rescaled according to second-order information in order to adjust for the geometry when optimizing in terms of distributions. Using the frameworks of Generalized Bayes and the Bayesian Learning Rule, we show that standard neural network optimizers can be derived as instances of natural-gradient descent given specific families of distributions. Finally, we derive practical natural-gradient-based optimization algorithms that closely resemble these standard neural network optimizers and can be used to train large neural networks effectively with arbitrary losses.

15.1 Natural-Gradient Descent

As introduced in Chapter 12, variational inference approximates a posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ over neural network parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ given data $\mathcal{D} = \{(x_t, y_t)\}_{t=1..N}$ by minimizing a regularized negative log-likelihood loss restricted to a (sub-)class of distributions $\mathcal{Q}$:

$$q_*(\boldsymbol{\theta}) = \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} \mathbb{E}_q[-\log p(\mathcal{D} \mid \boldsymbol{\theta})] + \mathbb{D}_{\text{KL}}(q \parallel p_0). \quad (15.1)$$

Here, p_0 is a prior belief over neural network weights discussed in detail in Chapter 28, and $\mathbb{D}_{\text{KL}}(q \parallel p_0)$ is the Kullback-Leibler divergence. For a well-specified model, and if $\mathcal{Q}$ is the space of all distributions, then the solution $q_*(\boldsymbol{\theta})$ is the Bayesian posterior [904, 1834]:

$$q_*(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{i=1}^N p(\mathcal{D}_i \mid \boldsymbol{\theta}), \quad (15.2)$$

here written factorized over individual data examples. This may initially seem surprising, because the objective is very similar to conventional deep learning objectives; the first term corresponds to data fit and the KL-divergence is similar to a regularizer. Later on, we will deepen this connection and show that many existing conventional deep learning optimizers actually solve a version of Equation 15.1 for a specific choice of $\mathcal{Q}$. We will also derive new optimizers of Equation 15.1 and generalize these methods to arbitrary losses beyond likelihood-based objectives. Then, we will use the term variational learning, as opposed to using the term variational inference. These new optimizers use natural-gradient descent but still have a similar form as conventional optimization algorithms.

The difference between natural-gradient and standard gradient descent is how they measure proximity. To illustrate this, we rewrite the objective from Equation 15.1 in terms of variational parameters $\boldsymbol{\lambda}$. Denoting these by $\boldsymbol{\lambda}$ we define the following loss:

$$\mathcal{L}(\boldsymbol{\lambda}) := \mathbb{E}_q[-\log p(\mathcal{D} \mid \boldsymbol{\theta})] + \mathbb{D}_{\text{KL}}(q \parallel p_0). \quad (15.3)$$

Intuitively, we might want to optimize this loss using stochastic gradient descent (SGD) on the parameters $\boldsymbol{\lambda}$, since such strategies are well-established and work well for deep learning. This strategy has been used in earlier

variational inference algorithms for neural networks, for example, Bayes-by-Backprop [174] which showed promising results on smaller neural network experiments. SGD then corresponds to using the following rule,

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\boldsymbol{\lambda}_t), \quad (15.4)$$

where the gradient $\nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\boldsymbol{\lambda}_t)$ may be based on mini-batches and the step-size is $\rho_t > 0$.

Following Khan and Rue [883, Sec. 2], it can be shown that Equation 15.4 solves the following objective:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg \min_{\boldsymbol{\lambda}} \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t), \boldsymbol{\lambda} \rangle + \frac{1}{2\rho_t} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_t\|_2^2, \quad (15.5)$$

which reveals a Euclidean penalty on the parameters $\boldsymbol{\lambda}_t$. However, as we show next, this may be a bad choice for comparing distributions, for example, to understand how close a $q(\boldsymbol{\theta})$ is to a previous iterate $q_t(\boldsymbol{\theta})$ or to the “true” posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$.

To illustrate this, we repeat a classical example from the literature [763, 1213] in which two pairs of Gaussian distributions are compared using the squared Euclidean distance. One pair, $\mathcal{N}(\mu_1 = 0, \sigma_1 = 0.01)$ and $\mathcal{N}(\mu_2 = 0.1, \sigma_2 = 0.01)$, is intuitively dissimilar: μ_2 is multiples of σ_1 and σ_2 away from μ_1 . Yet, the Euclidean distance of their parameters $\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|_2^2 = (\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2 = 0.01$ is small. The other pair consists of $\mathcal{N}(\mu_1 = 10, \sigma_1 = 1000)$ and $\mathcal{N}(\mu_2 = 0, \sigma_2 = 1000)$. Both have high variance and should intuitively be more similar but the Euclidean distance is 100. This motivates choosing an objective that measures proximity in terms of distributions. The example also shows that the resulting methods should perhaps be tied to second-order information, because the difference between distributions largely depends on their variances. Indeed, we will show in Section 15.3 that natural-gradient descent is tied to second-order methods.

Instead of the Euclidean distance, natural-gradient descent uses the Kullback-Leibler (KL) divergence to compare distributions. Say we are given an approximate posterior $q(\boldsymbol{\theta})$ and a prior $p_0(\boldsymbol{\theta})$. The KL divergence between them is then defined as:

$$\mathbb{D}_{\text{KL}}(q(\boldsymbol{\theta}) \parallel p_0(\boldsymbol{\theta})) := \mathbb{E}_q \left[\log \left(\frac{q(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta})} \right) \right] = \mathbb{E}_q[\log q(\boldsymbol{\theta})] - \mathbb{E}_q[\log p_0(\boldsymbol{\theta})], \quad (15.6)$$

where the first part is actually the negative of the entropy $\mathcal{H}(q) := -\mathbb{E}_q[\log q(\boldsymbol{\theta})]$ of q and the second one is called cross-entropy. Replacing the squared Euclidean distance with the KL finally results in the following [883, Sec. 2.1]:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg \min_{\boldsymbol{\lambda}} \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t), \boldsymbol{\lambda} \rangle + \frac{1}{\rho_t} \mathbb{D}_{\text{KL}}(q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \parallel q_t(\boldsymbol{\theta})), \quad (15.7)$$

By using the following quadratic expansion [1296],

$$\mathbb{D}_{\text{KL}}(q_{t+1}(\boldsymbol{\theta}) \parallel q_t(\boldsymbol{\theta})) \approx \frac{1}{2} (\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t)^T F(\boldsymbol{\lambda}_t) (\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t), \quad (15.8)$$

a new update rule can be derived that uses natural gradients:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t F(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\boldsymbol{\lambda}_t). \quad (15.9)$$

This update rule is the Bayesian learning rule [883] when optimizing likelihoods, given that $\boldsymbol{\lambda}$ are the natural parameters of exponential-family distributions, as we will show in Chapter 15.2. The natural gradients are the gradients preconditioned with the Fisher:

$$\tilde{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t) := F(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\boldsymbol{\lambda}_t). \quad (15.10)$$

The Fisher accounts for an interplay between the different dimensions of $\boldsymbol{\lambda}$ which becomes evident by writing it as the covariance of the score function of q as

$$F(\boldsymbol{\lambda}) = \text{Cov}_{q(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta})]. \quad (15.11)$$

As a consequence, natural-gradient-based variational inference can converge faster than regular gradient-descent-based methods when the Fisher is well-conditioned [1045]. Finally, the natural-gradient descent in Equation 15.9 is invariant to the parameterization, which does not always hold when using a Euclidean distance [1044].

Still, the inversion of the Fisher and the fact that it grows quadratically with the dimension of the neural network might seem problematic. In the next Section 15.2, we show that these issues are overcome by using exponential-family distributions. We first introduce the exponential family and then use exponential-family distributions to introduce practical second-order optimizers for natural-gradient descent.

15.1.1 Distinction from Natural Gradient Methods for Maximum Likelihood Estimation

Before that, we note that the natural gradient presented here is distinct from the natural gradient for maximum likelihood estimation of $p(\mathcal{D} \mid \boldsymbol{\theta})$. There, the idea is to use natural gradients to compare the output distributions of a neural network given a point estimate over parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ rather than to compare distributions over parameter configurations. The update is then only written in terms of $\boldsymbol{\theta}$ as:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \rho_t F(\boldsymbol{\theta}_t)^{-1} \nabla_{\boldsymbol{\theta}_t} [-\log p(y_t \mid x_t, \boldsymbol{\theta}_t)]. \quad (15.12)$$

This is often used for second-order optimization [26, 1135]. Then, the Fisher is defined based on data and the corresponding natural gradient is $F(\boldsymbol{\theta}_t)^{-1} \nabla_{\boldsymbol{\theta}_t} [-\log p(y_t \mid x_t, \boldsymbol{\theta}_t)]$. To make the distinction, [1845] also refer to this as *natural gradient for point estimation* and the natural gradient discussed here as *natural gradient for variational inference*.

15.2 Generalized Bayes & The Bayesian Learning Rule

In the previous section we have presented the variational objective for inference in Equation 15.1, discussed that it splits into data fit and regularizer terms, and shown how it can be optimized using natural-gradient-based methods. However, the data fit term in Equation 15.1 is defined assuming that the likelihood $p(\mathcal{D} \mid \boldsymbol{\theta})$ is well-specified in the sense that for some $\boldsymbol{\theta}^* \in \Theta$, we have that $p(\mathcal{D} \mid \boldsymbol{\theta}^*)$ equals the true data-generating distribution. In addition, it is assumed that this is what the practitioner wants to optimize. However, in practice it is often not the case. For example, in machine translation we might instead want to optimize a quality rating assigned to a translation by a human rater or in LLM post-training we might want to optimize some reward assigned to a generation of the LLM by another LLM. These motivate extending variational inference to what is often referred to as Generalized Bayes [904] or also called variational learning, where arbitrary losses can be used. The posterior is then called generalized posterior [164, 246, 1858] that can also be related to Gibbs posteriors [25] with connections to thermodynamics [246].

Let us first rewrite the posterior $q_*(\boldsymbol{\theta})$ from Equation 15.2 by using the loss function $\ell(\mathcal{D}_i, \boldsymbol{\theta}) = -\log p(\mathcal{D}_i \mid \boldsymbol{\theta})$ for a data likelihood $p(\mathcal{D}_i \mid \boldsymbol{\theta})$ as follows [904]:

$$q_*(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{i=1}^N \exp \{-\ell(\mathcal{D}_i, \boldsymbol{\theta})\}. \quad (15.13)$$

By using generic $\ell(\mathcal{D}_i, \boldsymbol{\theta})$ we obtain a generalized posterior. Following [164], Equation 15.13 is the solution to the following generalized Bayesian objective if we choose $\mathcal{Q}$ to be the space of all distributions on Θ :

$$q_*(\boldsymbol{\theta}) = \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} \mathbb{E}_q [\ell(\mathcal{D}, \boldsymbol{\theta})] + \mathbb{D}_{\text{KL}}(q \parallel p_0). \quad (15.14)$$

Again, $\ell : \mathcal{X} \times \mathcal{Y} \times \Theta \rightarrow \mathbb{R}$ can be any suitable loss function and for $\ell(\mathcal{D}, \boldsymbol{\theta}) \propto -\log p(\mathcal{D} \mid \boldsymbol{\theta})$ we recover the Bayesian posterior [1834]. In variational learning we restrict $\mathcal{Q}$ to a specific class of distributions but need to note that the optimal distribution $q_* \in \mathcal{Q}$ might not coincide with the true solution, as this true solution may not be contained in $\mathcal{Q}$. This is similar to what is referred to as model uncertainty in Hüllermeier and Waegeman [804, inter alia] when choosing a specific hypothesis space in general machine learning.

The objective in Equation 15.14 makes the parallels between standard optimization of neural networks and generalized variational methods more concrete: both optimize a loss term and a regularizer term but in Bayesian methods the loss uses an expectation over possible parameter values to account for uncertainty and a KL divergence to compare distributions, while standard optimization uses a loss based on a point estimate and Euclidean penalties. Intuitively, variational learning might seem to be more general and, indeed, we will next derive a Bayesian learning rule that allows recovering standard optimizers.

15.2.1 Exponential Family Distributions

We introduce the exponential-family distributions which include Gaussian, Poisson, and other distributions. We will show that exponential-family distributions are convenient for optimizing the generalized Bayesian objective in Equation 15.14.

Definition 15.1. *Minimal exponential family*

Given natural parameters $\boldsymbol{\lambda}$, sufficient statistics $\mathbf{T}(\boldsymbol{\theta})$, base measure $h(\boldsymbol{\theta})$ and log-partition function $Z(\boldsymbol{\lambda})$, a minimal exponential family distribution is defined as:

$$q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \exp[\boldsymbol{\lambda}^T \mathbf{T}(\boldsymbol{\theta}) - Z(\boldsymbol{\lambda})]. \quad (15.15)$$

The bijective map $\boldsymbol{\mu}(\boldsymbol{\lambda}) = \mathbb{E}_{q_{\boldsymbol{\lambda}}}[\mathbf{T}(\boldsymbol{\theta})]$ is called expectation parameter.

For example, for diagonal Gaussians of the form $\mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, \mathbf{s})$ the natural parameters are $\boldsymbol{\lambda} = (\mathbf{s}\mathbf{m}, -\frac{1}{2}\mathbf{s})^T$, where $\mathbf{s} := (\boldsymbol{\sigma}^2)^{-1}$ is called precision.

15.2.2 Bayesian Learning Rule

We will now describe the Bayesian learning rule (BLR) [883] which is a natural-gradient descent algorithm that optimizes Equation 15.14 using exponential family distributions. First, we rewrite Equation 15.14 by noting that, when choosing a prior $p_0 \propto \exp[-r(\boldsymbol{\theta})]$ proportional to a regularizer $r(\boldsymbol{\theta})$, we can express the KL term using the entropy $\mathcal{H}(q)$ as follows up to an additive constant of q that we drop:

$$\mathbb{D}_{\text{KL}}(q \| p_0) = \mathbb{E}_q[r(\boldsymbol{\theta})] - \mathcal{H}(q) + \text{const}. \quad (15.16)$$

This leads to the following objective, where we redefine $\bar{\ell}(\mathcal{D}, \boldsymbol{\theta}) := \sum_{i=1}^N \ell(\mathcal{D}_i, \boldsymbol{\theta}) + r(\boldsymbol{\theta})$:

$$q_*(\boldsymbol{\theta}) = \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} \mathbb{E}_q[\bar{\ell}(\mathcal{D}, \boldsymbol{\theta})] + \mathbb{D}_{\text{KL}}(q \| p_0) \quad (15.17)$$

$$= \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} \mathbb{E}_q[\bar{\ell}(\mathcal{D}, \boldsymbol{\theta})] - \mathcal{H}(q). \quad (15.18)$$

We can reinterpret the objective as balancing loss and entropy of q .

Using this formulation the BLR updates the natural parameters $\boldsymbol{\lambda}$ of an exponential family distribution as follows, where the natural gradient from Equation 15.9 is now defined in terms of an arbitrary loss with a standard deep learning regularizer:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t F(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}} [\mathbb{E}_{q_t}[\bar{\ell}(\mathcal{D}, \boldsymbol{\theta})] - \mathcal{H}(q)] \quad (15.19)$$

$$\Leftrightarrow \boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}(\boldsymbol{\lambda})} [\mathbb{E}_{q_t}[\bar{\ell}(\mathcal{D}, \boldsymbol{\theta})] - \mathcal{H}(q)]. \quad (15.20)$$

The two updates are equivalent because for exponential-family distributions we have that:

$$F(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_t}[\cdot] = \nabla_{\boldsymbol{\mu}(\boldsymbol{\lambda})} \mathbb{E}_{q_t}[\cdot]. \quad (15.21)$$

This simplifies computation as no inversion of the Fisher is required anymore due to the mapping between natural and expectation parameter [736]. The next important step is then to express the term $\nabla_{\boldsymbol{\mu}(\boldsymbol{\lambda})} \mathbb{E}_{q_t}[\cdot]$ in a way that is convenient for deep learning algorithms.

15.2.3 Stochastic Gradient Descent (SGD) from the Bayesian Learning Rule (BLR)

For Gaussian $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, \boldsymbol{\Sigma})$ the BLR update simplifies to [883, Eq. 6]:

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1 - \rho_t) \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}(\boldsymbol{\lambda})} \mathbb{E}_q[\bar{\ell}(\mathcal{D}, \boldsymbol{\theta}) + \log h(\boldsymbol{\theta})]. \quad (15.22)$$

To illustrate how the BLR can be used to recover and design optimization algorithms we briefly outline how SGD can be seen as an instance of Equation 15.20 by following the derivations from [883, Sec. 1.3.1]. The derivation starts by substituting an isotropic Gaussian posterior $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, I)$ in Equation 15.22. Then, we have that $\boldsymbol{\mu}(\boldsymbol{\theta}) = \mathbf{m} = \boldsymbol{\lambda}$ and $\mathbb{E}_q[\log h] = -\mathbf{m}^T \mathbf{m} / 2 + \text{const}$. As a consequence, Equation 15.20 reduces to:

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \nabla_{\mathbf{m}} \mathbb{E}_{q_t}[\bar{\ell}(\mathcal{D}_t, \mathbf{m}_t)] |_{\mathbf{m}=\mathbf{m}_t}, \quad (15.23)$$

where $\mathbf{m}_t$ are the model parameters at step t and the variance is kept fixed to identity. This is gradient descent but the gradient is smoothed via the expectation. To arrive at regular gradient descent we use the Delta method as detailed in [883, App. C], where an expectation is approximated by the expectation of its first-order Taylor expansion, i.e

$$\mathbb{E}_q[f(\boldsymbol{\theta})] \approx \mathbb{E}_q[f(\mathbf{m}) + (\boldsymbol{\theta} - \mathbf{m})^T \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\mathbf{m}}] = f(\mathbf{m}). \quad (15.24)$$

The delta method can be used to map from Bayesian methods to point estimation. Plugging this in Equation 15.23 we immediately get SGD. Next, we turn to a more practical part of the chapter and use similar kinds of derivations to derive NGVI optimizers that can work well even for large deep learning models.

15.3 Variational Online Newton Methods

In this section we discuss practical natural-gradient-based deep learning optimizers that can be derived from the Bayesian learning rule. These optimizers are connected to Newton's method as they use second-order information. The first method is often referred to as Variational Online Newton (VON) and makes use of a Gaussian approximation $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, \boldsymbol{\Sigma})$. Usually, a mean-field assumption (13) with diagonal Gaussians $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, \text{diag}(\boldsymbol{\sigma}^2))$ is made but we will be more general at least in the beginning.

We follow the derivations of Khan et al. [881, Sec. 3.1, App. C, App. D] and Khan and Rue [883] and present the main points. A factorized loss $\ell(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \ell(\mathcal{D}_i, \boldsymbol{\theta})$ leads to:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{\theta})}[N \cdot \ell(\mathcal{D}_i, \boldsymbol{\theta})] + \mathbb{D}_{\text{KL}}(q(\boldsymbol{\theta}) \parallel p_0(\boldsymbol{\theta})). \quad (15.25)$$

The size of $\mathcal{D}$ balances data fit and regularization which is intuitive as observing more data means that we can rely more on the data fit which we will later exploit for tuning.

Using the chain rule to express $\nabla_{\boldsymbol{\mu}}$ in terms of $\mathbf{m}$ and $\boldsymbol{\Sigma}$ we can rewrite Equation 15.22:

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} - 2\rho_t[\nabla_{\boldsymbol{\Sigma}} \mathcal{L}(\boldsymbol{\theta})] \quad (15.26)$$

$$\mathbf{m}_{t+1} = \mathbf{m}_t + \rho_t \boldsymbol{\Sigma}_{t+1} [\nabla_{\mathbf{m}} \mathcal{L}(\boldsymbol{\theta})]. \quad (15.27)$$

While this already resembles a practical algorithm, we need to handle the gradient of the expectation that is hidden inside $\mathcal{L}(\boldsymbol{\theta})$. Next, we introduce Bonnet's [187] and Price's theorem [1345], which lets us push the gradient inside the expectations [1264, 1391, inter alia].

Theorem 15.1. *Bonnet's theorem*

Let $f : \Theta \rightarrow \mathbb{R}$ once-differentiable, $q = \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$, and $\mathbb{E}_q[\|f(\boldsymbol{\theta})\|] < \infty$. Then, we have:

$$\nabla_{\mathbf{m}} \mathbb{E}_q[f(\boldsymbol{\theta})] = \mathbb{E}_q[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})]. \quad (15.28)$$

Theorem 15.2. *Price's theorem*

Let $f : \Theta \rightarrow \mathbb{R}$ twice-differentiable, $q = \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$, and $\mathbb{E}_q[\|f(\boldsymbol{\theta})\|] < \infty$. Then, we have:

$$\nabla_{\boldsymbol{\Sigma}} \mathbb{E}_q[f(\boldsymbol{\theta})] = \frac{1}{2} \mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})]. \quad (15.29)$$

Price's theorem shows that the variance is directly related to the Hessian.

Then we can write the updates as follows using $\delta > 0^1$:

$$\boldsymbol{\Sigma}_{t+1}^{-1} \leftarrow (1 - \rho_t) \boldsymbol{\Sigma}_t^{-1} + \rho_t [N \cdot \mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2 \ell(\mathcal{D}_t, \boldsymbol{\theta})] + \delta I] \quad (15.30)$$

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \boldsymbol{\Sigma}_{t+1} [N \cdot \mathbb{E}_q[\nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}_t, \boldsymbol{\theta})] + \delta \mathbf{m}_t], \quad (15.31)$$

The expectations $\mathbb{E}_q[\cdot]$ can be approximated using Monte-Carlo methods with samples $\boldsymbol{\theta}_t \sim \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}_t, \boldsymbol{\Sigma}_t)$ at each iteration. This is similar to standard neural network training but noise is added to the current learned mean $\mathbf{m}_t$ which induces parameter exploration.

In addition to the expectations, we need to find tractable approximations of the Hessian $H(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}}^2 \ell(\mathcal{D}_t, \boldsymbol{\theta}) \in \mathbb{R}^{P \times P}$. While a mean-field approximation using $\mathcal{N}(\boldsymbol{\theta}; \mathbf{m}_t, \text{diag}(\boldsymbol{\sigma}^2))$ reduces storage cost to the same as the cost of storing the mean, we still require an efficient computation of $\text{diag}(H(\boldsymbol{\theta}))$. For this, we use Stein's Lemma [1560] for multivariate Gaussians as presented by Lin et al. [1046, Lemma 5] and then apply it for Hessian computation.

Lemma 15.1. *Stein's Lemma for Multivariate Gaussians*

Given $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}, \boldsymbol{\Sigma})$, $f : \Theta \rightarrow \mathbb{R}$, continuous, locally ACL, and $\mathbb{E}_q[\|f(\boldsymbol{\theta})\|] < \infty$. Then, we have the following identity:

$$\mathbb{E}_q[\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \mathbf{m})f(\boldsymbol{\theta})] = \mathbb{E}_q[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})]. \quad (15.32)$$

As a consequence, we can rewrite Price's theorem as follows.

Theorem 15.3. *Price's Theorem - revisited*

Let $f : \Theta \rightarrow \mathbb{R}$ once-differentiable, $q = \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$, and $\mathbb{E}_q[\|f(\boldsymbol{\theta})\|] < \infty$. Then, we have:

$$\nabla_{\boldsymbol{\Sigma}} \mathbb{E}_q[f(\boldsymbol{\theta})] = \frac{1}{2} \mathbb{E}_q[\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \mathbf{m})\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})]. \quad (15.33)$$

¹In Khan et al. [881], $\delta = \delta'/N$, where δ' is the prior precision of $p_0 = \mathcal{N}(\boldsymbol{\theta}; 0, \mathbf{I}/\delta')$.

Now we only need to require f to be once-differentiable anymore. Similarly, we could also rewrite Bonnet's theorem to obtain a gradient estimator even for non-differentiable functions. By applying Price's theorem to the loss $\ell(\mathcal{D}, \theta)$ we get:

$$\mathbb{E}_q[\nabla_{\theta}^2 \ell(\mathcal{D}, \theta)] = \mathbb{E}_q[\Sigma^{-1}(\theta - \mathbf{m}) \nabla_{\theta} \ell(\mathcal{D}, \theta)]. \quad (15.34)$$

That is, the Hessian of the loss can be calculated by using first-order gradients. A Monte-Carlo estimator of Equation 15.34 is an unbiased estimate of the expected Hessian.

We next present VON with one Monte-Carlo sample and diagonal Gaussians. The variance in VON can be defined as $\sigma^2 := 1/[N(\mathbf{h} + \delta)]$ for a Hessian estimate $\mathbf{h}$ and weight-decay δ . The diagonal Hessian $\mathbf{h}$ is approximated by using Price's theorem coupled with Stein's Lemma which only requires first-order gradients. These gradients are estimated using an unbiased parameter sample from q_t . VON can then be implemented by the following algorithm for minibatch $\mathcal{B} \subset \mathcal{D}$, where we use the shorthand notation $\ell_i(\theta)$ for the loss of the i -th example in a mini-batch:

$$\hat{\mathbf{g}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta_t} \ell_i(\theta_t) \quad \text{where } \theta_t \sim q_t(\theta) \quad (15.35)$$

$$\hat{\mathbf{h}} \leftarrow (\theta_t - \mathbf{m}_t) \hat{\mathbf{g}} / \sigma_t^2 \quad (15.36)$$

$$\mathbf{g} \leftarrow \beta_1 \mathbf{g} + (1 - \beta_1) \hat{\mathbf{g}} \quad (15.37)$$

$$\mathbf{h}_{t+1} \leftarrow \beta_2 \mathbf{h}_t + (1 - \beta_2) \hat{\mathbf{h}} \quad (15.38)$$

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t (\mathbf{g} + \delta \mathbf{m}_t) / (\mathbf{h}_{t+1} + \delta) \quad (15.39)$$

$$\sigma_{t+1}^2 \leftarrow 1/N(\mathbf{h}_{t+1} + \delta) \quad (15.40)$$

The preconditioner in VON uses the Hessian without square root, which is used in Adam-like algorithms, and the Hessian is calculated online. Still, due to its similarity to conventional optimizers VON is easy to implement in standard deep learning frameworks.

However, so far there is no guarantee that the Hessian will stay positive in VON. If the Hessian is not positive, q is not in the family of diagonal Gaussians anymore, as inversion of the covariance is not possible. While the authors of [881] note that backtracking could be used to prevent the Hessian from becoming negative, this strategy is slow. This motivates the next sections on VOGN and IVON, where we will first use a different Hessian estimator that guarantees positivity, and then revisit VON with a positive definite correction that moves q back into the feasible set of Equation 15.18.

15.3.1 VOGN

To optimize the variational objective we need to ensure that the solutions remain in the feasible set consisting of valid distributions $q \in \mathcal{Q}$. In the Gaussian case this means that $\Sigma \in \mathbb{R}^{P \times P}$ has to be invertible. For VON, we have determined the relationship $\sigma^2 = 1/[N(\mathbf{h} + \delta)]$ between variance and Hessian but VON does not ensure that $\mathbf{h} + \delta$ stays positive. In this section we present the Variational Online Gauss-Newton (VOGN) method based on [881, 1266] which uses a mean-field Gaussian with a positive-definite Hessian approximation.

The approximation is based on the Generalized Gauss-Newton matrix (GGN). In order to define the GGN, let us decompose the Hessian by referring to the neural network output of a network with parameters $\theta \in \mathbb{R}^P$ as $f_{\theta}(x)$. The following result was obtained by [1473] and is also discussed in [1135, inter alia]. It states that the Hessian consists of:

$$\nabla_{\theta}^2 \ell_i(\theta) = \underbrace{(\nabla_{\theta} f_{\theta}(x_i))^T \nabla_f^2 \ell_i(\theta) \nabla_{\theta} f_{\theta}(x_i)}_{\text{GGN}} + \underbrace{\nabla_{\theta}^2 f_{\theta}(x_i) \nabla_z \ell(y_i, z)|_{z=f(x_i)}}_{\text{residual}}, \quad (15.41)$$

where $\nabla_{\theta} f_{\theta}(x_i)$ is the Jacobian. By dropping the second term in the decomposition we arrive at the GGN approximation of the Hessian. The GGN retains the curvature induced by the loss with respect to network outputs but drops curvature due to the network's nonlinear parameterization. For many settings it can be shown that the GGN and Fisher coincide. We refer the reader to [1135, Sec. 9.2] for a discussion.

Clearly, the GGN itself again requires the computation of a Hessian and Jacobian. This can be prohibitive in itself, as for a neural network with P parameters and C output classes we have that $\nabla_{\theta} f_{\theta}(x_i) \in \mathbb{R}^{P \times C}$. For example, for LLMs C is often in the order of tens of thousands as it consists of all possible tokens from the vocabulary. Conveniently, though, there are estimators for the GGN that are easy and comparably fast to compute simply based on first-order gradients. The estimator used in VOGN was presented in Graves [660] and simply

averages the squared gradients of examples in a mini-batch:

$$\hat{\mathbf{h}} \approx \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} [\nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta})]^2. \quad (15.42)$$

This estimator is guaranteed to be nonnegative but popular deep learning libraries directly calculate a sum of gradients over the mini-batch and the per-example gradients can only be obtained with additional computation and overhead. Indeed, [1510] report an overhead of almost 1.8x in an image classification setting when compared to the estimator used in VON. Hence, it might seem intuitive to just square the gradients of the minibatch directly, i.e., to move the average inside the square:

$$\hat{\mathbf{h}}^{\text{GM}} \approx \left[\frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}) \right]^2. \quad (15.43)$$

This is often referred to as gradient magnitude approximation [195] and the authors of [881] use this to derive a variational version of RMSprop called Vprop. Both estimates are compared in Khan et al. [881, Thm. 1] which we restate here next.

Theorem 15.4. *Bias of Gradient Magnitude Approximation*

Given are mini-batches $\mathcal{B}$ of size M drawn uniformly from $p(\mathcal{B})$ over all mini-batches that can be formed based on $\mathcal{D}$ of size N , GGN $\mathbf{h}$ calculated on all data, Gradient Magnitude approximation $\hat{\mathbf{h}}$, and gradients $\mathbf{g}$ on all data. Using $w = \frac{1}{M}(N - M)/(N - 1)$ we have:

$$\mathbb{E}_{p(\mathcal{B})}[\hat{\mathbf{h}}^{\text{GM}}] = w\mathbf{h} + (1 - w)\mathbf{g}^2. \quad (15.44)$$

Only $M = 1$ provides an unbiased estimator and the bias grows with M .

Similar to Osawa et al. [1266, Alg. 1], VOGN can then be implemented by simply replacing the Hessian estimator in VON with:

$$\hat{\mathbf{h}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} [\nabla_{\boldsymbol{\theta}_i} \ell_i(\boldsymbol{\theta}_t)]^2. \quad (15.45)$$

Using a strictly positive damping of the Hessian guarantees a strictly positive estimate. The authors of VOGN report promising results, reaching an accuracy that is close to or better than SGD and Adam [894] with improved calibration. However, they report a significant computational overhead of ca. 1.7x due to the per-example gradients required, similar to the overhead reported in Shen et al. [1510]. In many modern applications, where training can take days, weeks, and months, this may be too expensive. Next, we describe the Improved Variational Online Newton (IVON) optimizer [1047, 1510] which is fast because it uses the same estimator as in VON and guarantees that q stays within the feasible set.

15.3.2 shen2024variational

At the beginning of this section, we have introduced the VON algorithm as a practical optimizer for neural networks with a mean-field Gaussian posterior approximation. However, VON does not constrain q to be Gaussian which requires the covariance to be positive-definite (while $\mathbf{m}$ is unconstrained). The constraint can be unfulfilled when running VON on a non-convex loss $\ell(\mathcal{D}_i, \boldsymbol{\theta})$, because in Equation 15.30 the Hessian $\mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2 \ell(\mathcal{D}_t, \boldsymbol{\theta})]$ can be indefinite if ℓ is not convex. This is also true for Equation 15.40, where the reparameterization trick is used to estimate the Hessian.

IVON overcomes this by adding a positive-definite correction to the Hessian update, following the work of Lin et al. [1047]. We refer the reader to their work for a detailed derivation and only use their main result in the context of IVON next. In IVON every update step starts with a current approximate posterior $q_t = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}_t, \boldsymbol{\sigma}_t^2)$. IVON uses the following update rule for the Hessian, where the positive-definite correction is shown in grey:

$$\hat{\mathbf{g}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}_i} \ell_i(\boldsymbol{\theta}_t) \quad \text{where } \boldsymbol{\theta}_t \sim q_t(\boldsymbol{\theta}) \quad (15.46)$$

$$\hat{\mathbf{h}} \leftarrow \hat{\mathbf{g}} \cdot (\boldsymbol{\theta}_t - \mathbf{m}_t) / \boldsymbol{\sigma}_t^2 \quad (15.47)$$

$$\mathbf{h}_{t+1} \leftarrow \beta_2 \mathbf{h}_t + (1 - \beta_2) \hat{\mathbf{h}} + \frac{1}{2} (1 - \beta_2)^2 (\mathbf{h}_t - \hat{\mathbf{h}})^2 / (\mathbf{h}_t + \delta). \quad (15.48)$$

Algorithm 19: Improved Variational Online Newton (IVON)

Inputs : learning rates $\{\rho_t\}$; gradient momentum $\beta_1 \in [0, 1)$; Hessian momentum $\beta_2 \in [0, 1)$; weight decay $\delta > 0$; scaling parameter $\lambda > 0$; Hessian initialization $h_0 > 0$

Outputs: mean parameter $\mathbf{m}_{t+1}$ and variance σ_{t+1}^2

initialize $\mathbf{m} \leftarrow$ (NN weight init), $\mathbf{h} \leftarrow h_0 \cdot \mathbf{I}$, $\sigma^2 \leftarrow 1/(\lambda(\mathbf{h} + \delta))$, $\mathbf{g} \leftarrow 0$, $t \leftarrow 1$, and $\alpha_t \leftarrow (h_0 + \delta)\alpha_t$ for all t

repeat

$\hat{\mathbf{g}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta_t} \ell_i(\theta_t)$, where $\theta_t \sim q_t(\theta) = \mathcal{N}(\theta; \mathbf{m}_t, \text{diag}(\sigma_t^2))$

$\hat{\mathbf{h}} \leftarrow \hat{\mathbf{g}} \cdot (\theta_t - \mathbf{m}_t) / \sigma_t^2$

$\mathbf{g} \leftarrow \beta_1 \mathbf{g} + (1 - \beta_1) \hat{\mathbf{g}}$

$\mathbf{h} \leftarrow \beta_2 \mathbf{h}_t + (1 - \beta_2) \hat{\mathbf{h}} + \frac{1}{2}(1 - \beta_2)^2 (\mathbf{h}_t - \hat{\mathbf{h}})^2 / (\mathbf{h}_t + \delta)$

$\bar{\mathbf{g}} \leftarrow \mathbf{g} / (1 - \beta_1^t)$

$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \alpha_t (\bar{\mathbf{g}} + \delta \mathbf{m}_t) / (\mathbf{h} + \delta)$

$\sigma_{t+1}^2 \leftarrow 1/(\lambda(\mathbf{h} + \delta))$

until convergence

return $\mathbf{m}_{t+1}, \sigma_{t+1}^2$

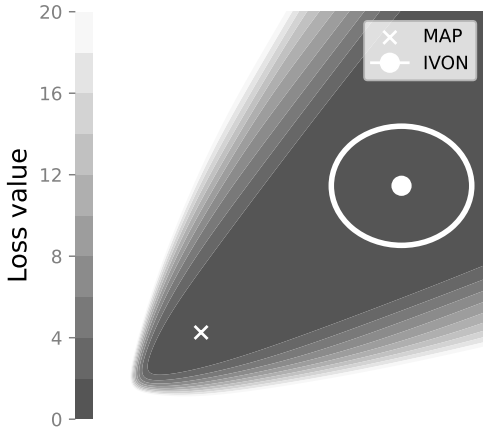


Figure 15.1: The solution found by IVON lands in a flat region away from the MAP. Figure reproduced from the code of Shen et al. [1510].

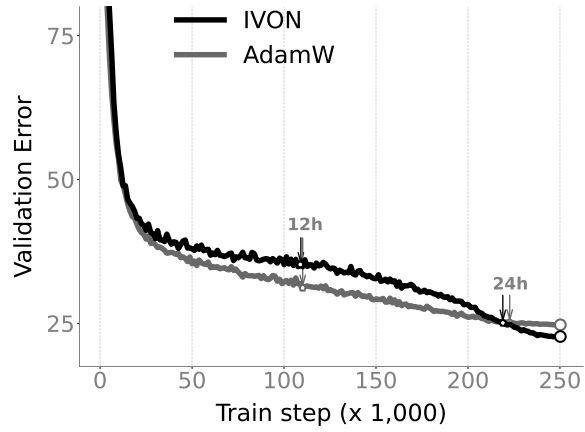


Figure 15.2: IVON vs. AdamW on ImageNet, taken from Shen et al. [1510] with permission.

The correction $\frac{1}{2}(1 - \beta_2)^2 (\mathbf{h}_t - \hat{\mathbf{h}})^2 / (\mathbf{h}_t + \delta) >^{\text{a.s.}} 0$ ensures that the current approximate posterior $q_{t+1} = \mathcal{N}(\theta; \mathbf{m}_{t+1}, \text{diag}(\sigma_{t+1}^2))$ is still a valid Gaussian distribution as $\mathbf{h}_{t+1}$ is always positive-definite if $\mathbf{h}_t$ is positive-definite, following [1047, Thm. 1]. A proof for a generalized setting with full covariance is found in [1047, App. E.1].

In addition to this, IVON uses a slightly different diagonal variance $\sigma^2 = 1/(\lambda(\mathbf{h} + \delta))$, where $\mathbf{h} = h_0 \cdot \mathbf{I}$ for $h_0 > 0$ in the first iteration, δ is a weight-decay parameter from using a prior with scalar variance and λ is the effective sample size that can be tuned as a hyperparameter. Tuning λ can be helpful in many situations, for example, when training many epochs on little data it can be set to $\lambda > N$ to simulate a setting where more data is available. Another example is when predicting using the posterior: when sampling from the final $q(\theta)$, it is possible to decrease λ to sample more diverse models (that are farther away from $\mathbf{m}$) or increase λ to sample less diverse models. We refer the reader to Chapter 43 for a more in-depth discussion of this in the context of large language models. The final updates of IVON are shown in Algorithm 19. IVON can also be used with multiple MC samples to estimate the gradient. In this case, the gradient is just averaged over MC samples and also estimate $\hat{\mathbf{h}}$ is averaged over multiple samples.

The noise in IVON can be helpful for regularizing training, because IVON constantly explores new solutions according to the chosen noise level during training. This leads to it “bouncing” away from sharp solutions, as adding even a bit of noise there will heavily increase the loss, towards flatter minima, as is illustrated in Figure 15.1 on a small logistic regression problem, where the maximum-a-posteriori (MAP) problem finds a sharp minimum but IVON finds a flat minimum. Flatter minima have long been connected to improved generalization and better

uncertainty estimates. Furthermore, gradient spikes can be smoothed out when multiple MC samples are used due to the Gaussian smoothing.

Altogether, IVON enables scaling natural-gradient descent with online Newton methods to large problems. Here, we show results on ImageNet for a ResNet-50 in Figure 15.2, which were obtained in Shen et al. [1510], and in Section 43.2 results for pretraining GPT-2 with 773M parameters from scratch are shown. In the ImageNet experiment, IVON obtains better or comparable accuracy as AdamW, improved calibration and uncertainty, and only requiring a computational overhead of 1.0-1.2x (depending on model size) due to the additional parameter sampling step. This shows that variational learning of large models is feasible and due to the techniques shown in this chapter is also easy to implement and use.

15.4 Concluding Remarks

We have presented natural-gradient-descent algorithms to learn generalized posterior using the framework of variational learning, where arbitrary losses can be used. In order to derive these algorithms, we have introduced the Bayesian Learning Rule as well as various convenient techniques to, for example, estimate second-order information. We have introduced various algorithms that can use these techniques to learn distributions over neural network parameters in an efficient way with similar computational requirements as conventional neural network optimizers.

Chapter 16

Particle-Based Variational Inference

16.1 Introduction

Bayesian deep learning (BDL) seeks to estimate the posterior distribution over the parameters of a deep neural network. In this setting, however, traditional methods for performing inference face significant challenges.

Markov chain Monte Carlo (MCMC) methods like Hamiltonian Monte Carlo (HMC) (Chapter 1) are asymptotically exact and can, in principle, generate samples from the true posterior. In practice, however, they are often prohibitively slow and scale poorly to large datasets or high-dimensional neural network models.

Variational inference (VI) offers a more scalable alternative by reframing inference as an optimisation problem (Chapter 12): one posits an approximate distribution $q(\theta)$, and adjusts its parameters to minimise the Kullback–Leibler (KL) divergence $D_{\text{KL}}(q\|\pi)$ to the target posterior $\pi(\theta)$ [170, 1840]. The most common form of VI used in Bayesian neural networks is mean-field VI (Chapter 13), which assumes all weights are independent (e.g., diagonal Gaussian posteriors). While computationally convenient, this approximation is highly restrictive: it fails to capture complex posterior correlations or multimodality, often underestimates uncertainty, and can lead to miscalibrated models [1108]. More expressive variational families (e.g., full-covariance Gaussians, normalising flows) do exist, but they introduce substantial optimisation challenges and are hard to tune, often requiring model-specific derivations.

If classical MCMC algorithms are asymptotically accurate but too slow for deep learning, whereas mean-field VI is fast but often leads to crude posterior approximations, then is there a way of leveraging the computational efficiency of VI with the nonparametric nature of MCMC? *Particle-Based Variational Inference* (ParVI) methods aim to bridge this gap by combining the strengths of both MCMC and variational inference.

The core idea, first introduced by Liu and Wang [1064], is to maintain a set of n particles $\{\theta^i\}_{i=1}^n$ that collectively represent an approximate posterior. Instead of evolving each particle via random Monte Carlo moves, ParVI (in its prototypical form) deterministically transports the entire set of particles towards regions of high posterior density [1064]. In each iteration, the particles are used to approximate a population transport direction that decreases the KL divergence to the target π , via a form of gradient descent in the space of probability distributions [1064]. By using information from the target’s score function (the gradient of $\log \pi(\theta)$) and coupling the particles through a repulsive interaction, ParVI methods encourage particles to spread out and cover the posterior’s support rather than collapsing to a single mode (see Figure 16.1 for a visual illustration). In effect, ParVI treats Bayesian inference as an optimisation in function space, finding a distribution (represented by particles) that minimises $D_{\text{KL}}(q\|\pi)$. This approach retains the flexibility of nonparametric sampling, since the empirical distribution represented by the particles is not constrained to lie within a specific parametric family (e.g., Gaussian), while leveraging gradient-based optimisation for efficiency and scalability. That said, for a finite number of particles the empirical approximation is itself restricted, and its quality depends on the number of particles, the choice of kernel, and the interactions between the particles.

Mathematical Background and Reading Guide. Relative to the preceding chapters, parts of this chapter make use of some additional mathematical background, in particular reproducing kernel Hilbert spaces [e.g., 128], and notions from optimal transport and gradient flows on the space of probability measures [e.g., 27, 292, 1458, 1679]. To help the reader navigate the chapter, sections marked with an asterisk (*) are more advanced, and can safely be skipped on a first reading: a reader interested primarily in the basic methodology can read Sections 16.2, 16.3, 16.6, and 16.7, returning to the starred sections later.

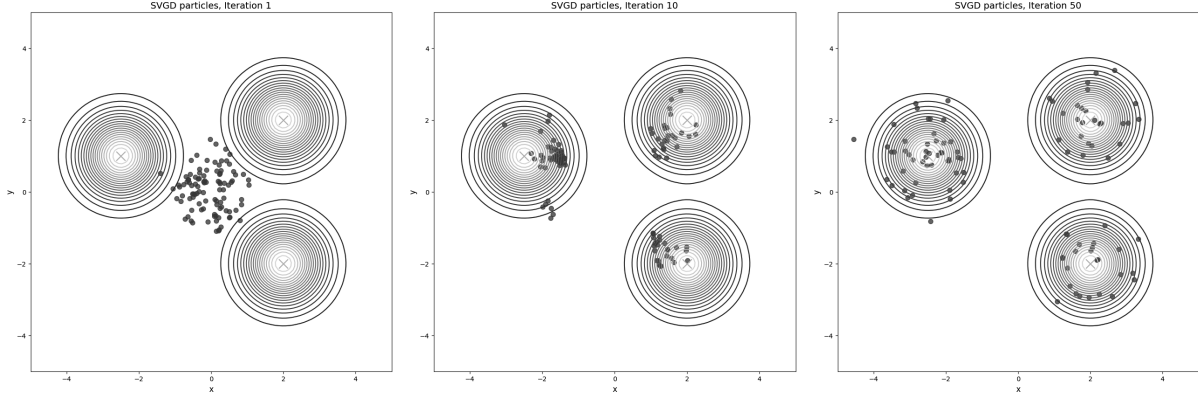


Figure 16.1: An illustration of a particle-based variational inference approach on a mixture of three Gaussians target distribution π . Using the Stein variational gradient descent algorithm (SVGD, Algorithm 20), 100 particles are simulated from $\mathcal{N}(0, 0.5 I)$ and evolved via the SVGD dynamics; the three panels show the particles after 1, 10, and 50 iterations (left to right). Initially, the particles are attracted to regions of high posterior density, then the repulsive force of the kernel encourages the particles to spread out and produce a nonparametric approximation of the posterior distribution.

16.2 Variational Objectives Revisited

16.2.1 The KL Divergence and the Evidence Lower Bound (ELBO)

VI frames Bayesian posterior estimation as an optimisation problem. In particular, given a target posterior $\pi(\theta) := p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$, VI seeks an approximate distribution $q(\theta)$ that minimises the KL divergence to the target posterior:

$$D_{\text{KL}}(q\|\pi) = \mathbb{E}_q \left[\log \frac{q(\theta)}{\pi(\theta)} \right] = -\mathbb{E}_q [\log \pi(\theta)] + \mathbb{E}_q [\log q(\theta)]. \quad (16.1)$$

Minimising the KL divergence is equivalent to maximising the evidence lower bound (ELBO), since

$$D_{\text{KL}}(q\|\pi) = -\underbrace{\mathbb{E}_q [\log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D})]}_{:=\mathbb{E}_q [\log \pi(\theta)]} + \mathbb{E}_q [\log q(\theta)] \quad (16.2)$$

$$= -\underbrace{\mathbb{E}_q [\log p(\mathcal{D}|\theta) + \log p(\theta) - \log q(\theta)]}_{:=\text{ELBO}(q)} + \log p(\mathcal{D}) \quad (16.3)$$

$$= -\text{ELBO}(q) + \text{const.} \quad (16.4)$$

The ELBO provides a tractable objective for fitting the variational approximation. In conventional VI [170], q is constrained to belong to a parametric family, say $\{q_\psi\}_{\psi \in \Psi}$, with variational parameters ψ . For example, one might specify a Gaussian family of distributions, in which case the variational parameters are simply $\psi = [\mu, \Sigma]$ specifying the mean μ and covariance Σ of the variational family. One then maximises the ELBO with respect to the variational parameters (e.g., via stochastic gradient ascent) to obtain the approximation of the posterior.

In ParVI, we still seek a variational distribution $q(\theta)$ that maximises the ELBO. However, the constraint that $q(\theta)$ must belong to a parametric family is now relaxed. Thus, we must now solve an *infinite-dimensional* optimisation problem over the space of probability measures, rather than a *finite-dimensional* optimisation problem over the space of the variational parameters.

16.2.2 Infinitesimal Transport of q by a Vector Field

Let q and π be strictly positive C^1 densities on $\mathbb{R}^d$. Fix a C^1 vector field $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with sufficient decay (or compact support) so that the boundary terms below vanish. For ε small, we consider a transport map obtained as a small perturbation of the identity map, viz

$$T_\varepsilon(\theta) = \theta + \varepsilon \phi(\theta), \quad T_0 = \text{Id}. \quad (16.5)$$

We will consider distributions obtained as the push-forward of q under this transport map, viz

$$q_\varepsilon = (T_\varepsilon)_\# q, \quad \text{i.e.} \quad \int f(\theta) q_\varepsilon(\theta) d\theta = \int f(T_\varepsilon(\theta)) q(\theta) d\theta \quad \forall f. \quad (16.6)$$

By the change-of-variables formula, the density q_ε is given explicitly by

$$q_\varepsilon(\theta) = q(T_\varepsilon^{-1}(\theta)) \left| \det \nabla T_\varepsilon^{-1}(\theta) \right|, \quad (16.7)$$

where T_ε^{-1} is the inverse of T_ε , and where $\nabla T_\varepsilon^{-1}$ denotes the Jacobian matrix of T_ε^{-1} . A first-order expansion in ε gives the classical *continuity equation*

$$\left. \frac{d}{d\varepsilon} \right|_{\varepsilon=0} q_\varepsilon(\theta) = -\nabla_\theta \cdot (q(\theta)\phi(\theta)) \iff \partial_\varepsilon q_\varepsilon + \nabla \cdot (q_\varepsilon \phi) = 0 \quad \text{at } \varepsilon = 0. \quad (16.8)$$

16.2.3 The Derivative of the KL Divergence

We would like to define the transport map T_ε , via the vector field ϕ , in such a way that it decreases the KL divergence: $D_{\text{KL}}((T_\varepsilon)_\# q \| \pi) \leq D_{\text{KL}}(q \| \pi)$. In order to identify a suitable vector field, we now differentiate the KL functional w.r.t. the perturbation magnitude ε , viz

$$D_{\text{KL}}(q_\varepsilon \| \pi) = \int q_\varepsilon(\theta) \log \frac{q_\varepsilon(\theta)}{\pi(\theta)} d\theta. \quad (16.9)$$

Using the continuity equation in (16.8) and the product rule, we can compute this as follows:

$$\begin{aligned} \left. \frac{d}{d\varepsilon} D_{\text{KL}}(q_\varepsilon \| \pi) \right|_{\varepsilon=0} &= \int \left. \frac{d}{d\varepsilon} \right|_{\varepsilon=0} \left\{ q_\varepsilon(\theta) \log \frac{q_\varepsilon(\theta)}{\pi(\theta)} \right\} d\theta \\ &= \int \left[-\nabla \cdot (q(\theta)\phi(\theta)) \right] \left(\log \frac{q(\theta)}{\pi(\theta)} + 1 \right) d\theta \\ &= \int q(\theta)\phi(\theta) \cdot \nabla_\theta \left(\log \frac{q(\theta)}{\pi(\theta)} + 1 \right) d\theta \quad (\text{integration by parts}) \\ &= \int q(\theta)\phi(\theta) \cdot (\nabla \log q(\theta) - \nabla \log \pi(\theta)) d\theta \\ &= \mathbb{E}_q \left[\phi(\theta)^\top (\nabla \log q(\theta) - \nabla \log \pi(\theta)) \right] \\ &= -\mathbb{E}_q \left[\nabla_\theta \cdot \phi(\theta) + \phi(\theta)^\top \nabla \log \pi(\theta) \right], \end{aligned}$$

where in the final line we have used the identity $\mathbb{E}_q[\nabla \cdot \phi] = -\mathbb{E}_q[\phi^\top \nabla \log q]$. We have thus shown that

$$\left. \frac{d}{d\varepsilon} D_{\text{KL}}(q_\varepsilon \| \pi) \right|_{\varepsilon=0} = -\mathbb{E}_q [T_\pi \phi(\theta)], \quad T_\pi \phi := \nabla \cdot \phi + \phi^\top \nabla \log \pi. \quad (16.10)$$

16.2.4 The Direction of Steepest Descent

From the previous expression, any direction that *increases* $\mathbb{E}_q[T_\pi \phi]$ yields a *decrease* in $D_{\text{KL}}(\cdot \| \pi)$. Naturally, we would like to choose the direction ϕ which maximally decreases the KL divergence. That is, we would like to solve

$$\arg \min_{\phi \in \mathcal{G}} \left\{ \left. \frac{d}{d\varepsilon} D_{\text{KL}}((T_\varepsilon)_\# q \| \pi) \right|_{\varepsilon=0} : \|\phi\|_{\mathcal{G}} \leq 1 \right\} = \arg \max_{\phi \in \mathcal{G}} \left\{ \mathbb{E}_q [T_\pi \phi(\theta)] : \|\phi\|_{\mathcal{G}} \leq 1 \right\}, \quad (16.11)$$

where $\mathcal{G}$ is a normed space of functions. In order to make progress, we will consider the restriction of the vector field ϕ to a ball within a reproducing-kernel Hilbert space (RKHS). This will allow us to obtain a closed-form expression for the vector field which maximises the negative gradient on the RHS of (16.11). We begin by providing some background on the operator T_π .

The Stein Operator and Stein's Identity. The operator T_π is known as the (Langevin) *Stein operator* [1559], due to its connection with *Stein's identity*. Let π be a strictly positive C^1 density on $\mathbb{R}^d$ with score $u_\pi(\theta) := \nabla \log \pi(\theta)$. Then, for any sufficiently regular test function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfying mild zero boundary conditions, Stein's identity states that

$$\mathbb{E}_\pi[T_\pi f(\theta)] = \mathbb{E}_\pi[\nabla_\theta \cdot f(\theta) + f(\theta)^\top u_\pi(\theta)] = 0, \quad (16.12)$$

which is easily seen to hold via integration by parts. We emphasise that the (Langevin) Stein operator T_π depends only on the *score* and therefore does not require the normalising constant of π . Indeed, $\nabla_\theta \log \pi = \nabla_\theta \log \tilde{\pi}$ for $\pi = \tilde{\pi}/Z$. This is a particularly useful property in Bayesian deep learning, where the normalising constant is intractable.

The Stein Discrepancy. The identity in (16.12) characterises π in the sense that, under mild conditions on the class of test functions $\mathcal{G}$, $\mathbb{E}_q[T_\pi f] = 0$ for all such $f \in \mathcal{G}$ if and only if $q = \pi$ a.e. [e.g., 652, 1559]. More generally, we can use the maximum value of this expectation over all test functions $f \in \mathcal{G}$ to characterise the difference between the distributions q and π . In particular, we define the *Stein discrepancy* as

$$\mathbb{D}(q\|\pi) := \max_{f \in \mathcal{G}} \{\mathbb{E}_q[T_\pi f] : \|f\|_{\mathcal{G}} \leq 1\}. \quad (16.13)$$

Clearly, the choice of the function class $\mathcal{G}$ here is critical. It must be both sufficiently rich to ensure that $\mathbb{D}(q\|\pi)$ has adequate discriminative power, while not too rich to render $\mathbb{D}(q\|\pi)$ computationally intractable. Following Chwialkowski et al. [307], Gorham and Mackey [652], Liu and Wang [1064], we will take $\mathcal{G}$ to be a vector-valued RKHS. This ensures not only that the linear functional $f \mapsto \mathbb{E}_q[T_\pi f]$ is continuous, but also that the optimisation problem in (16.11), (16.13) has a closed form.

The Kernel Stein Discrepancy. Let $\mathcal{H}$ be a RKHS of scalar-valued functions with positive-definite C^2 kernel $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Let $\mathcal{H}^d := \mathcal{H} \times \cdots \times \mathcal{H}$ denote the corresponding $\mathbb{R}^d$ -valued RKHS, equipped with inner product $\langle f, g \rangle_{\mathcal{H}^d} = \sum_{\ell=1}^d \langle f_\ell, g_\ell \rangle_{\mathcal{H}}$. Using the reproducing property of the Hilbert space, and integration by parts, it holds for all $f \in \mathcal{H}^d$ that

$$\begin{aligned} \mathbb{E}_q[T_\pi f] &= \mathbb{E}_q[\nabla \cdot f(\theta)] + \mathbb{E}_q[f(\theta)^\top u_\pi(\theta)] \\ &= \mathbb{E}_q[\text{Tr}(\nabla_\theta f(\theta))] + \mathbb{E}_q[f(\theta)^\top u_\pi(\theta)] \\ &= \langle f(\cdot), \mathbb{E}_{\theta \sim q}[k(\theta, \cdot) u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)] \rangle_{\mathcal{H}^d} = \langle f, \mu_{q,\pi} \rangle_{\mathcal{H}^d}. \end{aligned} \quad (16.14)$$

where in the final line we have used $\mu_{q,\pi}$ to denote the *Stein witness function*, defined as

$$\mu_{q,\pi}(\cdot) := \mathbb{E}_{\theta \sim q}[k(\theta, \cdot) u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)] \in \mathcal{H}^d. \quad (16.15)$$

Substituting back into (16.13), the *kernel Stein discrepancy* (KSD) between q and π is given by the RKHS dual norm [307, 652, 1064]

$$\text{KSD}(q\|\pi) := \max_{f \in \mathcal{H}^d} \{\mathbb{E}_q[T_\pi f] : \|f\|_{\mathcal{H}^d} \leq 1\} = \|\mu_{q,\pi}\|_{\mathcal{H}^d}. \quad (16.16)$$

Using (16.15) and the reproducing property, we can obtain a closed-form expression for the (squared) KSD as

$$\text{KSD}^2(q\|\pi) = \mathbb{E}_{(\theta, \theta') \sim q \otimes q}[k_\pi(\theta, \theta')], \quad (16.17)$$

where

$$k_\pi(\theta, \theta') := u_\pi^\top(\theta) k(\theta, \theta') u_\pi(\theta') + u_\pi^\top(\theta) \nabla_{\theta'} k(\theta, \theta') + \nabla_\theta^\top k(\theta, \theta') u_\pi(\theta') + \text{Tr}(\nabla_\theta \nabla_{\theta'} k(\theta, \theta'))$$

is the *Stein kernel*. Given a set of particles $\{\theta^i\}_{i=1}^n \sim q$, a (biased) V-statistic estimator for the KSD is then given by

$$\widehat{\text{KSD}^2(q\|\pi)} = \frac{1}{n^2} \sum_{i,j=1}^n k_\pi(\theta^i, \theta^j), \quad (16.18)$$

with $\mathcal{O}(n^2)$ cost. As in the case of the maximum mean discrepancy (MMD) [667], an unbiased U-statistic variant is also available [1066]. Given i.i.d. samples from q , (16.18) is a biased estimator of $\text{KSD}^2(q\|\pi)$; for an interacting particle system, the same expression is more naturally interpreted as the squared KSD of the empirical measure $q^n = \frac{1}{n} \sum_{i=1}^n \delta_{\theta^i}$.

The KSD has a number of special properties which make it particularly useful in the context of Bayesian deep learning:

- **Normalisation-free.** Only $u_\pi = \nabla \log \pi$ is needed from the target π and the normalising constant Z cancels.
- **Convergence-determining.** For suitable kernels (e.g., the inverse multi-quadric (IMQ) kernel with appropriate bandwidth) and mild tail conditions on π , $\text{KSD}(q\|\pi) \rightarrow 0$ implies $q \Rightarrow \pi$ [307, 652].
- **Algorithmic role.** By (16.16), KSD quantifies the norm of the steepest-descent direction within $\mathcal{H}^d$. Its Riesz representer $\mu_{q,\pi}$ is exactly the vector field used by SVGD.

The Kernel Stein Discrepancy as the Direction of Steepest Descent. Returning to (16.16), we see that the maximiser of the constrained optimisation problem in (16.11) is given by the normalised witness function

$$\bar{\phi}_{q,\pi} = \arg \max_{\phi \in \mathcal{H}^d} \{\mathbb{E}_q[T_\pi \phi] : \|\phi\|_{\mathcal{H}^d} \leq 1\} = \frac{\mu_{q,\pi}}{\|\mu_{q,\pi}\|_{\mathcal{H}^d}}, \quad (16.19)$$

assuming $\mu_{q,\pi} \neq 0$; the corresponding maximum value is precisely $\text{KSD}(q\|\pi) = \|\mu_{q,\pi}\|_{\mathcal{H}^d}$. This characterises the direction of steepest descent of the KL divergence in the vector-valued RKHS $\mathcal{H}^d$. In practice, the SVGD algorithm introduced in the next section uses the *unnormalised* witness function, $\phi_{q,\pi}^* := \mu_{q,\pi}$, which points in the same direction; with this choice, the instantaneous rate of decrease of the KL divergence is $-\|\mu_{q,\pi}\|_{\mathcal{H}^d}^2 = -\text{KSD}^2(q\|\pi)$.

16.3 Stein Variational Gradient Descent (SVGD)

16.3.1 Basic Overview

In Section 16.2 we established that, given an infinitesimal transport map $T_\varepsilon(\theta) = \theta + \varepsilon \phi(\theta)$, which we recall satisfies

$$\left. \frac{d}{d\varepsilon} \text{D}_{\text{KL}}((T_\varepsilon)_\# q\|\pi) \right|_{\varepsilon=0} = -\mathbb{E}_q[T_\pi \phi(\theta)], \quad T_\pi \phi := \nabla \cdot \phi + \phi^\top \nabla \log \pi, \quad (16.20)$$

the direction of *steepest descent* that maximises the negative gradient in (16.20) amongst all perturbations lying in the ball $\{\phi \in \mathcal{H}^d : \|\phi\|_{\mathcal{H}^d} \leq 1\}$ of the vector-valued RKHS $\mathcal{H}^d$, is proportional to the witness function $\mu_{q,\pi}$, cf. (16.15). Following standard practice, we will work with the unnormalised direction

$$\phi_{q,\pi}^*(\cdot) = \mathbb{E}_{\theta \sim q} [k(\theta, \cdot) u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)]. \quad (16.21)$$

This result suggests the following iterative procedure for transforming an initial measure q_0 into the target posterior distribution π : for $k = 0, 1, \dots$

$$q_{k+1} = (T_{\varepsilon_k}^*)_\# q_k, \quad \text{where } T_{\varepsilon_k}^*(\theta) = \theta + \varepsilon_k \phi_{q_k,\pi}^*(\theta). \quad (16.22)$$

We can provide an equivalent description of (16.22) at the level of a single particle. In particular, for all $k = 0, 1, \dots$, we have $q_k = \text{Law}(\theta_k)$, where $(\theta_k)_{k \in \mathbb{N}}$ are defined according to¹

$$\theta_{k+1} = \theta_k + \varepsilon_k \mathbb{E}_{\theta \sim q_k} [k(\theta, \theta_k) u_\pi(\theta) + \nabla_\theta k(\theta, \theta_k)], \quad \theta_0 \sim q_0. \quad (16.23)$$

We identify two terms in the expectation in (16.23), each of which will play a different role in the updates:

$$\underbrace{\mathbb{E}_{\theta \sim q_k} [k(\theta, \theta_k) \nabla_\theta \log \pi(\theta)]}_{\text{attraction to high } \pi} + \underbrace{\mathbb{E}_{\theta \sim q_k} [\nabla_\theta k(\theta, \theta_k)]}_{\text{repulsion \& diversity}}. \quad (16.24)$$

The first term drives particles towards regions of high probability by following a smoothed version of the gradient of the log-density. The second term is a repulsive force which prevents all particles from collapsing to a single point.

¹Note a slight overload of notation: $k(\cdot, \cdot)$ denotes the kernel, while the subscript k indexes the iteration. The intended meaning will always be clear from context.

Functional Gradient Descent Interpretation. The updates defined by (16.22) can be viewed as a form of functional gradient descent in the vector-valued RKHS $\mathcal{H}^d$ [1064, Theorem 3.3]. Recall that, for any functional $\mathcal{F} : \mathcal{H}^d \rightarrow \mathbb{R}$, the *first variation* $D\mathcal{F}(f)(g)$ at $f \in \mathcal{H}^d$ along $g \in \mathcal{H}^d$ can be defined as

$$D\mathcal{F}(f)(g) := \lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} (\mathcal{F}(f + \varepsilon g) - \mathcal{F}(f)). \quad (16.25)$$

Meanwhile, assuming $\mathcal{F}$ is Fréchet differentiable, its *functional gradient* $\nabla_f \mathcal{F}(f) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the element in $\mathcal{H}^d$ such that, for all $g \in \mathcal{H}^d$:²

$$D\mathcal{F}(f)(g) = \langle \nabla_f \mathcal{F}(f), g \rangle_{\mathcal{H}^d}. \quad (16.27)$$

Consider the transport map $T_f = \text{id} + f$, where id is the identity map, and $f \in \mathcal{H}^d$. Define the functional

$$\mathcal{F}_{q,\pi}(f) = D_{\text{KL}}(T_{f\#}q \parallel \pi). \quad (16.28)$$

Then it is possible to show that the functional gradient $\nabla_f \mathcal{F}_{q,\pi}(f)$, evaluated at the zero map $f = 0$, satisfies [1064, Theorem 3.3]

$$\nabla_f \mathcal{F}_{q,\pi}(0) = -\phi_{q,\pi}^*. \quad (16.29)$$

Thus, the transport maps $T_{\varepsilon_k}^*(\theta)$ used to define the sequence of distributions $(q_k)_{k \in \mathbb{N}}$ in (16.22), viz

$$T_{\varepsilon_k}^*(\theta) = \theta + \varepsilon_k \phi_{q_k,\pi}^*(\theta) = \theta - \varepsilon_k \nabla_f \mathcal{F}_{q_k,\pi}(0)(\theta) \quad (16.30)$$

can be thought of as performing a sequence of steps of functional gradient descent in the RKHS.

Continuous-Time Limit. The updates in (16.22) and (16.23) can also be viewed as a time-discretisation of a continuous-time process referred to as the *Stein variational gradient flow*. In particular, as the step-size $\varepsilon_k \rightarrow 0$, the evolution of the densities $(q_k)_{k \in \mathbb{N}}$ defined by (16.22) is described by the continuity equation [1063, Section 3.3]

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta) \phi_{q_t,\pi}^*(\theta)), \quad \phi_{q_t,\pi}^*(\cdot) = \mathbb{E}_{\theta \sim q_t} [k(\theta, \cdot) u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)]. \quad (16.31)$$

Equivalently, the evolution of the particle $(\theta_k)_{k \in \mathbb{N}}$ defined by (16.23) is described by the nonlinear ODE

$$d\theta_t = \phi_{q_t,\pi}^*(\theta_t) dt, \quad \phi_{q_t,\pi}^*(\cdot) = \mathbb{E}_{\theta \sim q_t} [k(\theta, \cdot) u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)]. \quad (16.32)$$

One can view the updates in (16.22) or (16.23) as a forward Euler discretisation of the dynamics in (16.31) and (16.32), respectively.

Basic Convergence Results. The idea behind (16.22) or (16.23) is that, for sufficiently small step sizes $(\varepsilon_k)_{k \in \mathbb{N}_0}$, these updates decrease the KL divergence [1063, Theorem 3.3]. Under additional assumptions on the target, kernel, and limiting dynamics, this descent can be used to establish convergence to π ; see Section 16.8. Indeed, under suitable assumptions, it is possible to show that [918, Proposition 5]

$$D_{\text{KL}}(q_{k+1} \parallel \pi) - D_{\text{KL}}(q_k \parallel \pi) \leq -\varepsilon_k \text{KSD}^2(q_k \parallel \pi) + \mathcal{O}(\varepsilon_k^2). \quad (16.33)$$

Thus, for a sufficiently small step size ε_k , the KL divergence between the current distribution q_k and the target distribution π does indeed decrease at each iteration.

16.3.2 Particle-Based Approximation

In the k^{th} iteration, (16.23) requires computing an expectation with respect to the (unknown) distribution q_k . We will thus need to approximate this term. In order to do so, we can use an empirical approximation. In particular, let $\{\theta_0^i\}_{i=1}^n$ denote a set of particles drawn i.i.d. from the initial distribution q_0 . We will then update, for $k = 0, 1, 2, \dots$, and $i = 1, \dots, n$,

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \mathbb{E}_{\theta \sim q_k^n} [k(\theta, \theta_k^i) u_\pi(\theta) + \nabla_\theta k(\theta, \theta_k^i)], \quad (16.34)$$

²Equivalently:

$$\mathcal{F}(f + \varepsilon g) = \mathcal{F}(f) + \varepsilon \langle \nabla_f \mathcal{F}(f), g \rangle_{\mathcal{H}^d} + \mathcal{O}(\varepsilon^2). \quad (16.26)$$

Algorithm 20: Stein Variational Gradient Descent (SVGD)

Inputs : target score $u_\pi(\theta) = \nabla \log \pi(\theta)$; kernel k ; step sizes $\{\varepsilon_k\}_{k \in \mathbb{N}}$; particles $\{\theta_0^i\}_{i=1}^n$
Outputs: updated particles $\{\theta_K^i\}_{i=1}^n$
for $k = 0$ **to** $K - 1$ **do**
 compute scores $g_j \leftarrow u_\pi(\theta_k^j)$ for $j = 1, \dots, n$
 for $i = 1$ **to** n **do**
 $\phi^i \leftarrow \frac{1}{n} \sum_{j=1}^n \left[k(\theta_k^j, \theta_k^i) g_j + \nabla_{\theta_k^j} k(\theta_k^j, \theta_k^i) \right]$
 $\theta_{k+1}^i \leftarrow \theta_k^i + \varepsilon_k \phi^i$
return $\{\theta_K^i\}_{i=1}^n$

where $q_k^n = \frac{1}{n} \sum_{i=1}^n \delta_{\theta_k^i}$ denotes the empirical distribution of the particles $\{\theta_k^i\}_{i=1}^n$ in the k^{th} iteration. That is, for $k = 0, 1, 2, \dots$ and $i = 1, \dots, n$,

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \frac{1}{n} \sum_{j=1}^n \left[k(\theta_k^j, \theta_k^i) u_\pi(\theta_k^j) + \nabla_{\theta_k^j} k(\theta_k^j, \theta_k^i) \right]. \quad (16.35)$$

This algorithm, summarised in Algorithm 20, is known as *Stein variational gradient descent* (SVGD).

16.3.3 Complexity and Practical Implementation

Computational Complexity. For typical choices of the kernel k , each iteration in (16.35) requires the computation of all pairwise interactions. The computational cost thus scales as $\mathcal{O}(n^2)$ w.r.t. the number of kernel (and kernel gradient) evaluations, and as $\mathcal{O}(n)$ w.r.t. the number of score function evaluations. Given a large number of particles, or large datasets, there are several ways to reduce the computational cost:

- **Mini-batch scores.** Estimate the score function $u_\pi(\theta) := \nabla_\theta \log \pi(\theta)$ using mini-batches of the data. This yields a stochastic but unbiased approximation (cf. Section 16.6).
- **Kernel approximations.** Use random Fourier features to reduce the $\mathcal{O}(n^2)$ interaction to $\mathcal{O}(nm)$ or $\mathcal{O}(n \log n)$ [1065]; see Section 16.6.3 for further details.
- **Parallelism.** Use parallel computing, noting that both the kernel matrix and sums are data-parallel and GPU-friendly.

Bandwidth Selection. The choice of the kernel bandwidth critically affects the balance between the attractive and repulsive terms in the update equation, and thus the dispersal of particles. Adaptive schedules (e.g., the median heuristic) are commonly employed.

Step Sizes and Stopping Criteria. As with standard gradient descent, the step sizes $(\varepsilon_k)_{k \in \mathbb{N}}$ must also be chosen with some care: if too large, the discretisation of the underlying gradient flow becomes unstable; if too small, convergence is unnecessarily slow. In practice, a small constant step size, or an adaptive scheme such as AdaGrad, is often used [1064]. Alternatively, learning-rate-free variants of SVGD, based on ideas from parameter-free online optimisation (so-called coin betting), remove the need to select a step-size schedule altogether [1503], and can be extended to constrained domains [1506]. More recently, adaptive step-size schedules derived from the gradient flow perspective (see Section 16.4) have been obtained for SVGD and related sampling algorithms, with convergence guarantees matching those of their optimally tuned counterparts up to logarithmic factors [1504]. Regarding termination, there is no universally agreed stopping criterion: common choices include running for a fixed computational budget, or monitoring the magnitude of the particle updates or an empirical estimate of the KSD, cf. (16.18), and stopping when these fall below a threshold.

16.4 Particle-Based Variational Inference (ParVI) as Gradient Flow*

In Section 16.3, we derived the SVGD algorithm, and showed how it could be viewed as a time discretisation, combined with a particle approximation of the evolving measure, of a continuous-time, mean-field process known

as the Stein variational gradient flow. In this section, we show that this is just one example of a *gradient flow over the space of probability measures*. We provide a general recipe for constructing such gradient flows, and describe several examples, including the Stein variational gradient flow as a special case. We also show how discretising these gradient flows in time, and approximating the evolving measure with particles, yields practical ParVI algorithms.

16.4.1 Basics of Gradient Flows

In this section, we introduce the notion of a gradient flow over the space of probability measures. Our exposition relies on the formal identification of the space of probability measures with an (infinite-dimensional) Riemannian manifold. Thus, we begin by recalling some basic concepts from Riemannian geometry.

Manifold, Tangent Space, Cotangent Space. For convenience, in this section we will consider the manifold of smooth, strictly positive densities over $\mathbb{R}^d$, viz $\mathcal{P} = \{q \in C^\infty(\mathbb{R}^d) : \int q d\theta = 1, q > 0\}$. For each $q \in \mathcal{P}$, the *tangent space* of $\mathcal{P}$ is given by $\mathcal{T}_q\mathcal{P} \subseteq \{\sigma \in C^\infty(\mathbb{R}^d) : \int \sigma d\theta = 0\}$. We will write $\mathcal{T}_q^*\mathcal{P}$ for the cotangent space, the topological dual of $\mathcal{T}_q\mathcal{P}$.

Metric Tensor, Onsager Operator, Riemannian Metric. Let $M(q) : \mathcal{T}_q\mathcal{P} \rightarrow \mathcal{T}_q^*\mathcal{P}$ denote a *metric tensor*, which maps elements in the tangent space $\mathcal{T}_q\mathcal{P}$ to elements in the cotangent space $\mathcal{T}_q^*\mathcal{P}$. We can then define a *Riemannian metric* $g_q : \mathcal{T}_q\mathcal{P} \times \mathcal{T}_q\mathcal{P} \rightarrow \mathbb{R}$ according to $g_q(\sigma_1, \sigma_2) = \langle M(q)\sigma_1, \sigma_2 \rangle$, where $\langle \cdot, \cdot \rangle$ is the *duality pairing* from $\mathcal{T}_q^*\mathcal{P} \times \mathcal{T}_q\mathcal{P} \rightarrow \mathbb{R}$. The inverse of the metric tensor $M(q)$, denoted $M^{-1}(q) : \mathcal{T}_q^*\mathcal{P} \rightarrow \mathcal{T}_q\mathcal{P}$, is referred to as the *Onsager operator*.

First Variation, Riemannian Gradient. We next recall the definition of the *first variation* of $\mathcal{F}$ w.r.t. q . This is the unique function in $\mathcal{T}_q^*\mathcal{P}$ such that

$$\lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} [\mathcal{F}(q + \varepsilon\sigma) - \mathcal{F}(q)] = \left\langle \frac{\delta\mathcal{F}(q)}{\delta q}, \sigma \right\rangle, \quad (16.36)$$

for all $\sigma \in \mathcal{T}_q\mathcal{P}$. The gradient of $\mathcal{F}$ w.r.t. the Riemannian metric g_q , denoted $\text{grad}\mathcal{F}$, is then defined via the condition

$$g_q(\text{grad}\mathcal{F}, \sigma) = \left\langle \frac{\delta\mathcal{F}}{\delta q}, \sigma \right\rangle, \quad (16.37)$$

for all $\sigma \in \mathcal{T}_q\mathcal{P}$. Using the characterisation of the Riemannian metric in terms of the Onsager operator (see above), it is possible to write the Riemannian gradient as $\text{grad}\mathcal{F}(q) = M^{-1}(q) \frac{\delta\mathcal{F}}{\delta q}$.

Gradient Flows. We are now ready to define the *gradient flow* of $\mathcal{F}$ over the space $(\mathcal{P}, g)$. It is a curve $t \mapsto q_t$ that satisfies

$$\frac{\partial q_t}{\partial t} = -\text{grad}\mathcal{F}(q_t) := -M^{-1}(q_t) \frac{\delta\mathcal{F}}{\delta q} \Big|_{q=q_t}. \quad (16.38)$$

Provided the flow is well posed, and under additional assumptions on the functional $\mathcal{F}$ and the geometry (e.g., suitable convexity, coercivity, or functional inequalities; see below), one can often establish convergence of (16.38) to a minimiser of the functional $\mathcal{F}$; for the KL objective considered below, the target π is this minimiser, i.e., $q_t \rightarrow \pi$ as $t \rightarrow \infty$. This gradient flow thus provides us with a general method for transporting an initial reference distribution q_0 to a target posterior distribution π . In particular, we obtain different algorithms for sampling from the target posterior distribution according to different choices of:

1. the *objective functional* $\mathcal{F}(q)$ (e.g., KL divergence, other f -divergence, KSD);
2. the *geometry* over $\mathcal{P}$ (e.g., Wasserstein, Stein, Fisher-Rao);
3. the *time-discretisation* (e.g., forward Euler, backward Euler, forward flow);
4. the *space-discretisation* (e.g., finite element, particle-based approximation).

The Objective Functional. In order to simplify our discussion, we will assume throughout this section that the objective functional is the KL divergence: $\mathcal{F}(q) = \text{D}_{\text{KL}}(q \parallel \pi)$. In this case, we can calculate the first variation of $\mathcal{F}$ with respect to q as [e.g., 27]

$$\frac{\delta \mathcal{F}}{\delta q}(\theta) = \log \frac{q(\theta)}{\pi(\theta)} + 1 = \log q(\theta) - \log \pi(\theta) + 1. \quad (16.39)$$

This choice is well motivated: among all f -divergences with continuously differentiable f defined on the positive reals, the KL divergence is the only one whose first variation is unchanged (up to constants) if we scale π by a positive constant [282, Theorem 2.2].

Other choices of objective functional are, of course, possible, and lead to different ParVI algorithms for sampling from π . These include the kernel Stein discrepancy [919], the mollified interacting energy (MIE) [1030], and the χ^2 -divergence [293].

16.4.2 Wasserstein Gradient Flow

The Geometry. We begin by considering the gradient flow induced by the quadratic Wasserstein metric. In order to define the Wasserstein metric tensor, let us first define ψ_σ as the solution of the elliptic PDE

$$-\nabla_\theta \cdot (q \nabla_\theta \psi_\sigma) = \sigma. \quad (16.40)$$

The Wasserstein metric tensor $M(q) : \mathcal{T}_q \mathcal{P} \rightarrow \mathcal{T}_q^* \mathcal{P}$ and its inverse $M^{-1}(q) : \mathcal{T}_q^* \mathcal{P} \rightarrow \mathcal{T}_q \mathcal{P}$ are then defined according to [1272, 1679]

$$M(q)\sigma = \psi_\sigma, \quad \text{for all } \sigma \in \mathcal{T}_q \mathcal{P} \quad (16.41)$$

$$M^{-1}(q)\psi = -\nabla_\theta \cdot (q \nabla_\theta \psi), \quad \text{for all } \psi \in \mathcal{T}_q^* \mathcal{P}. \quad (16.42)$$

Meanwhile, following some elementary calculations, we identify the Wasserstein-Riemannian metric $g_q : \mathcal{T}_q \mathcal{P} \times \mathcal{T}_q \mathcal{P} \rightarrow \mathbb{R}$ as

$$g_q(\sigma_1, \sigma_2) = \int \nabla_\theta \psi_{\sigma_1}(\theta)^\top \nabla_\theta \psi_{\sigma_2}(\theta) q(\theta) d\theta. \quad (16.43)$$

The Gradient Flow. Substituting (16.42) into (16.38), and rearranging, we see that the Wasserstein gradient flow is given by the continuity equation

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta) v_t(\theta)), \quad v_t(\theta) = -\nabla_\theta \left. \frac{\delta \mathcal{F}}{\delta q} \right|_{q=q_t}(\theta). \quad (16.44)$$

In addition, substituting the first-variation of the KL divergence from (16.39) into (16.44) yields the Wasserstein gradient flow of the KL divergence as

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot \left(q_t(\theta) \nabla_\theta \log \frac{\pi(\theta)}{q_t(\theta)} \right) \quad (16.45)$$

$$= -\nabla_\theta \cdot (q_t(\theta) (\nabla_\theta \log \pi(\theta) - \nabla_\theta \log q_t(\theta))). \quad (16.46)$$

Formally, we can also provide a description of the gradient flow in (16.46) at the level of a single particle, the so-called *Lagrangian description*. In particular, for all $t \geq 0$, it holds that $q_t = \text{Law}(\theta_t)$, where $(\theta_t)_{t \geq 0}$ is the solution of

$$\dot{\theta}_t = \nabla_\theta \log \pi(\theta_t) - \nabla_\theta \log q_t(\theta_t), \quad \theta_0 \sim q_0. \quad (16.47)$$

This equation, sometimes referred to as the *deterministic Langevin equation* [675], is *non-linear* in the sense that the time evolution depends not only on the process θ_t , but also on its distribution q_t .

Alternatively, observe that the second term in (16.46) is equal to $\Delta_\theta q_t$, where Δ_θ denotes the Laplacian. We can thus rewrite (16.46) as

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta) \nabla_\theta \log \pi(\theta)) + \Delta_\theta q_t(\theta). \quad (16.48)$$

This is precisely the *Fokker-Planck equation* describing the time evolution of the marginal law of the overdamped Langevin dynamics w.r.t. π . In other words, for all $t \geq 0$, it holds that $q_t = \text{Law}(\theta_t)$, where now $(\theta_t)_{t \geq 0}$ is defined as the solution of the *stochastic differential equation*

$$d\theta_t = \nabla \log \pi(\theta_t) dt + \sqrt{2} dw_t, \quad \theta_0 \sim q_0, \quad (16.49)$$

where $w = (w_t)_{t \geq 0}$ is a standard $\mathbb{R}^d$ -valued Brownian motion. Thus, one can view the overdamped Langevin dynamics as the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric [852].

Convergence Rate. It is straightforward to obtain the dissipation of the KL divergence along the WGF. In particular, using the chain rule, we have that

$$\frac{d}{dt} D_{\text{KL}}(q_t \| \pi) = - \int q_t(\theta) \|\nabla_{\theta} \log \frac{q_t(\theta)}{\pi(\theta)}\|^2 d\theta = -\mathcal{I}(q_t \| \pi) \leq 0, \quad (16.50)$$

where $\mathcal{I}(q \| \pi)$ is the *relative Fisher information* of q with respect to π . Thus, $D_{\text{KL}}(q_t \| \pi)$ dissipates at a rate equal to the relative Fisher information. If, in addition, the target distribution π satisfies a log-Sobolev inequality with constant λ , then this inequality implies exponential convergence to the target distribution, at a rate determined by λ : $D_{\text{KL}}(q_t \| \pi) \leq e^{-2\lambda t} D_{\text{KL}}(q_0 \| \pi)$ [e.g., 292].

Practical Algorithms. In the previous section, we obtained two different differential equations, cf. (16.47) and (16.49), one of them stochastic, both of whose marginal distributions follow the Wasserstein gradient flow of the KL divergence, cf. (16.46). These equations suggest different algorithms for sampling from the target distribution π .

We first consider (16.49), which easily yields a practical sampling algorithm. In particular, using an Euler-Maruyama discretisation, we can define $(\theta_k)_{k=0,1,2,\dots}$ according to the update equation

$$\theta_{k+1} = \theta_k + \varepsilon_k \nabla_{\theta} \log \pi(\theta_k) + \sqrt{2\varepsilon_k} z_k, \quad \theta_0 \sim q_0, \quad (16.51)$$

where $(\varepsilon_k)_{k=0,1,\dots}$ is a positive, non-increasing sequence of step sizes; and where $(z_k)_{k=0,1,2,\dots}$ are i.i.d. standard normal random variables. The algorithm in (16.51) is nothing more than the *unadjusted Langevin algorithm* (ULA), also known as *Langevin Monte Carlo* (LMC), a popular MCMC algorithm [472]; its stochastic gradient variants are discussed in detail in Chapter 2, and in the recent monograph of Fearnhead et al. [509]. Despite its connection with the reformulation of sampling as an optimisation problem, this is not typically viewed as a “particle-based variational inference” algorithm, since it does not necessitate the use of an ensemble of particles.

Now consider (16.47). In this case, a forward Euler discretisation defines a sequence $(\theta_k)_{k=0,1,\dots}$ according to

$$\theta_{k+1} = \theta_k + \varepsilon_k (\nabla_{\theta} \log \pi(\theta_k) - \nabla_{\theta} \log q_k(\theta_k)), \quad \theta_0 \sim q_0. \quad (16.52)$$

One cannot implement this directly, since it depends on the unknown sequence of distributions $(q_k)_{k=0,1,\dots}$. Instead, one option is to evolve an ensemble of particles $(\theta_k^i)_{k=0,1,\dots}^{i=1,\dots,n}$, and update

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \left(\nabla_{\theta} \log \pi(\theta_k^i) - \widehat{\nabla_{\theta} \log q_k}(\theta_k^i) \right), \quad \theta_0^i \stackrel{i.i.d.}{\sim} q_0, \quad (16.53)$$

where $\widehat{\nabla_{\theta} \log q_k}$ denotes an approximation of $\nabla_{\theta} \log q_k$, formed using the particles $\{\theta_k^i\}_{i=1}^n$. There are various options for this approximation, including approximating the density q_k directly using a kernel-density estimate or a normalising flow [675], and approximating the score $\nabla \log q_k$ using score matching [179].

16.4.3 Stein Gradient Flow

We now provide a more rigorous interpretation of the mean-field, continuous-time limit of the SVGD algorithm, cf. (16.31), as a gradient flow over the space of probability measures equipped with a suitably chosen Riemannian metric.

The Geometry. Similar to before, cf. (16.40), to define the Stein metric tensor we will first introduce ψ_{σ} as the solution of the integro-partial differential equation

$$-\nabla_{\theta} \cdot \left(q(\theta) \int k(\theta', \theta) \nabla_{\theta'} \psi_{\sigma}(\theta') q(\theta') d\theta' \right) = \sigma(\theta), \quad (16.54)$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a symmetric positive-definite kernel. We can then define the *Stein metric tensor* and its inverse as [282, 462]

$$M(q)\sigma = \psi_{\sigma}, \quad \text{for all } \sigma \in \mathcal{T}_q \mathcal{P} \quad (16.55)$$

$$M^{-1}(q)\psi = -\nabla_{\theta} \cdot \left(q(\theta) \int k(\theta', \theta) \nabla_{\theta'} \psi(\theta') q(\theta') d\theta' \right), \quad \text{for all } \psi \in \mathcal{T}_q^* \mathcal{P}. \quad (16.56)$$

It follows that the Stein-Riemannian metric associated with this metric tensor is given by

$$g_q(\sigma_1, \sigma_2) = \int \nabla_{\theta} \psi_{\sigma_1}^{\top}(\theta) \left[\int k(\theta', \theta) \nabla_{\theta'} \psi_{\sigma_2}(\theta') q(\theta') d\theta' \right] q(\theta) d\theta. \quad (16.57)$$

The Gradient Flow. Similar to before, substituting (16.56) into (16.38), and rearranging, we see that the Stein gradient flow is given by the continuity equation

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta) v_t(\theta)), \quad v_t(\theta) = - \int k(\theta', \theta) \nabla_{\theta'} \left. \frac{\delta \mathcal{F}}{\delta q} \right|_{q=q_t}(\theta') q_t(\theta') d\theta'. \quad (16.58)$$

In addition, substituting the first-variation of the KL divergence from (16.39) into (16.58) yields the Stein gradient flow of the KL divergence (i.e., the Stein variational gradient flow) as

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot \left(q_t(\theta) \int k(\theta', \theta) \nabla_{\theta'} \log \frac{\pi(\theta')}{q_t(\theta')} q_t(\theta') d\theta' \right) \quad (16.59)$$

$$= -\nabla_\theta \cdot \left(q_t(\theta) \int k(\theta', \theta) [\nabla_{\theta'} \log \pi(\theta') - \nabla_{\theta'} \log q_t(\theta')] q_t(\theta') d\theta' \right). \quad (16.60)$$

Finally, assuming appropriate boundary conditions on k , and integrating by parts, we arrive back at

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta) \phi_{q_t, \pi}^*(\theta)), \quad \phi_{q_t, \pi}^*(\cdot) = \mathbb{E}_{\theta \sim q_t} [k(\theta, \cdot) \nabla_\theta \log \pi(\theta) + \nabla_\theta k(\theta, \cdot)]. \quad (16.61)$$

Convergence Rate. Using an appropriate version of the chain rule, one can show that the dissipation of the KL divergence along the Stein variational gradient flow is given by [918, Proposition 1]

$$\frac{d}{dt} D_{\text{KL}}(q_t \| \pi) = -\langle \phi_t, \mu_{q_t, \pi} \rangle_{\mathcal{H}^d} = -\|\mu_{q_t, \pi}\|_{\mathcal{H}^d}^2 = -\mathcal{I}_{\text{Stein}}(q_t \| \pi) := -\text{KSD}^2(q_t \| \pi) \leq 0. \quad (16.62)$$

Thus, $D_{\text{KL}}(q_t \| \pi)$ is non-increasing, and dissipates at a rate equal to the *Stein-Fisher information*, i.e., the squared KSD. Similar to before, if π satisfies a *Stein log-Sobolev inequality* with constant λ , then one has exponential convergence: $D_{\text{KL}}(q_t \| \pi) \leq D_{\text{KL}}(q_0 \| \pi) e^{-2\lambda t}$.

16.4.4 Fisher-Rao and Wasserstein-Fisher-Rao Gradient Flow

The Wasserstein geometry and the Stein geometry both result in gradient flows which *transport mass*. On the other hand, the Fisher-Rao geometry leads to a gradient flow which changes mass locally through birth-death dynamics rather than spatial transport; informally, mass can be removed in one region and created in another, as if it were *teleported*.

The Geometry. The Fisher-Rao Riemannian metric $g_q : \mathcal{T}_q \mathcal{P} \times \mathcal{T}_q \mathcal{P} \rightarrow \mathbb{R}$ is defined according to

$$g_q(\sigma_1, \sigma_2) = \int \frac{\sigma_1 \sigma_2}{q} d\theta, \quad (16.63)$$

or, writing $\sigma = q\psi_\sigma$,

$$g_q(\sigma_1, \sigma_2) = \int \psi_{\sigma_1} \psi_{\sigma_2} q d\theta. \quad (16.64)$$

Meanwhile the Fisher-Rao metric tensor $M(q) : \mathcal{T}_q \mathcal{P} \rightarrow \mathcal{T}_q^* \mathcal{P}$ associated to g_q , and its inverse $M^{-1}(q) : \mathcal{T}_q^* \mathcal{P} \rightarrow \mathcal{T}_q \mathcal{P}$, are given by

$$M(q)\sigma = \psi_\sigma = \frac{\sigma}{q}, \quad \text{for all } \sigma \in \mathcal{T}_q \mathcal{P} \quad (16.65)$$

$$M^{-1}(q)\psi = q(\psi - \mathbb{E}_q[\psi]), \quad \text{for all } \psi \in \mathcal{T}_q^* \mathcal{P}. \quad (16.66)$$

Strictly speaking, cotangent potentials are identified up to additive constants; equivalently, one may work with the centred representative $\psi - \mathbb{E}_q[\psi]$, in which case $M(q)$ and $M^{-1}(q)$ are strict inverses.

The Gradient Flow. It follows, substituting (16.66) into (16.38), that the Fisher-Rao gradient flow reads

$$\frac{\partial q_t(\theta)}{\partial t} = -q_t(\theta) \left(\left. \frac{\delta \mathcal{F}}{\delta q} \right|_{q=q_t}(\theta) - \mathbb{E}_{\theta \sim q_t} \left[\left. \frac{\delta \mathcal{F}}{\delta q} \right|_{q=q_t}(\theta) \right] \right). \quad (16.67)$$

Thus, substituting also (16.39), the Fisher-Rao gradient flow of the KL divergence is given by

$$\frac{\partial q_t(\theta)}{\partial t} = q_t(\theta) \left[\log \pi(\theta) - (\log q_t(\theta)) - \mathbb{E}_{\theta \sim q_t} [\log \pi(\theta) - \log q_t(\theta)] \right]. \quad (16.68)$$

This gradient flow is an example of a *birth-death process*. Since there is no spatial derivative, it can be viewed as an infinite system of ordinary differential equations, indexed by the parameter. The advantage of such a process is that it allows global moves (i.e., teleportation) of the mass of the probability density (e.g., from one mode to another) according to their relative weights, without the need to transport this mass through low probability regions. The disadvantage is that, due to the absence of diffusion, the support of q_t never changes during the evolution. This means that effective numerical simulation of the Fisher-Rao gradient flow is not straightforward.

One solution is to combine the Wasserstein and Fisher-Rao geometries into a hybrid geometry, which incorporates both mass transport and teleportation. In particular, the gradient flow w.r.t. the *Wasserstein-Fisher-Rao* geometry [1087] reads

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_{\theta} \cdot (q_t(\theta) v_t(\theta)) + q_t(\theta) \alpha_t(\theta), \quad (16.69)$$

where

$$v_t(\theta) = -\nabla_{\theta} \frac{\delta D_{\text{KL}}(q_t \parallel \pi)}{\delta q_t} = \nabla \log \pi(\theta) - \nabla \log q_t(\theta) \quad (16.70)$$

$$\begin{aligned} \alpha_t(\theta) &= - \left(\frac{\delta D_{\text{KL}}(q_t \parallel \pi)}{\delta q_t} - \mathbb{E}_{q_t} \left[\frac{\delta D_{\text{KL}}(q_t \parallel \pi)}{\delta q_t} \right] \right) \\ &= (\log \pi(\theta) - \log q_t(\theta)) - \mathbb{E}_{q_t} [\log \pi(\theta) - \log q_t(\theta)]. \end{aligned} \quad (16.71)$$

Convergence Rate. Under mild conditions on π and q_0 , it is possible to show that the asymptotic convergence rate of the Wasserstein-Fisher-Rao gradient flow is *independent* of the target distribution π . In particular, assuming q_0 and π both have bounded second moments, and that their ratio satisfies a certain growth condition, it holds that $D_{\text{KL}}(q_t \parallel \pi) \leq C e^{-t}$ for all $t \geq t_0$ [283, Theorem 4.1].

16.5 Second-Order Methods and Riemannian ParVI*

In Section 16.3, we saw that SVGD can be viewed as performing a form of functional gradient descent w.r.t. the KL divergence on a vector-valued RKHS. In particular, defining the functional

$$\mathcal{F}_{q,\pi}(f) = D_{\text{KL}}((\text{id} + f)_{\#} q \parallel \pi), \quad f \in \mathcal{H}^d, \quad (16.72)$$

then the (infinite-population version) of the SVGD algorithm was defined via the updates

$$q_{k+1} = (T_{\varepsilon_k}^*)_{\#} q_k, \quad \text{where } T_{\varepsilon_k}^* = \text{id} - \varepsilon_k \nabla_f \mathcal{F}_{q_k, \pi}(0). \quad (16.73)$$

Based on this perspective of SVGD, there are various ways to accelerate the SVGD algorithm. These include:

- Incorporating second-order information so that steps adapt to local ill-conditioning of the posterior.
- Adapting the *geometry* of the parameter manifold via a Riemannian (natural-gradient) metric.

We outline both of these approaches below.

16.5.1 Stein Variational Newton (SVN)

The Stein variational Newton (SVN) algorithm, introduced by Detommaso et al. [417], incorporates second-order information in the updates to accelerate convergence. In particular, the steepest descent direction in (16.73) is replaced by an approximation of the Newton direction.

Recall that the Newton direction is defined as the minimiser of a local quadratic (rather than linear) approximation of $\mathcal{F}$. In order to define this minimiser, we must first introduce the *second variation* $D^2 \mathcal{F}(0)(g, g')$ of $\mathcal{F}$ at 0, along the pair of directions $g, g' \in \mathcal{H}^d$, defined as

$$D^2 \mathcal{F}(0)(g, g') := \lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} (D\mathcal{F}(\varepsilon g')(g) - D\mathcal{F}(0)(g)). \quad (16.74)$$

Meanwhile, assuming $\mathcal{F}$ is twice Fréchet differentiable at 0, its *functional Hessian* $\nabla_f^2 \mathcal{F}(0)$ at 0 is the linear operator $\mathcal{H}^d \rightarrow \mathcal{H}^d$ such that, for all $g, g' \in \mathcal{H}^d$,

$$D^2 \mathcal{F}(0)(g, g') = \langle \nabla_f^2 \mathcal{F}(0)[g], g' \rangle_{\mathcal{H}^d}. \quad (16.75)$$

In the k^{th} iteration, we would like to find the Newton direction $g = (g_1, \dots, g_d) \in \mathcal{H}^d$ for the functional $\mathcal{F}_{q_k, \pi}(f)$, evaluated at $f = 0$. This is characterised by the first-order stationarity conditions: for all $g' \in \mathcal{H}^d$,

$$D^2 \mathcal{F}_{q_k, \pi}(0)(g, g') = -D \mathcal{F}_{q_k, \pi}(0)(g') \quad (16.76)$$

or, equivalently, for all $g' \in \mathcal{H}^d$,

$$\langle \nabla_f^2 \mathcal{F}_{q_k, \pi}(0)[g], g' \rangle_{\mathcal{H}^d} = -\langle \nabla_f \mathcal{F}_{q_k, \pi}(0), g' \rangle_{\mathcal{H}^d}. \quad (16.77)$$

Due to Liu and Wang [1064, Theorem 3.3] and Detommaso et al. [417, Theorem 1], we have explicit expressions for both the functional gradient $\nabla_f \mathcal{F}_{q_k, \pi}(0)$ and the functional Hessian $\nabla_f^2 \mathcal{F}_{q_k, \pi}(0)$, evaluated at $f = 0$. In particular, component-wise,

$$[\nabla_f \mathcal{F}_{q_k, \pi}(0)]_i(\theta') = -\mathbb{E}_{\theta \sim q_k} [k(\theta, \theta') \partial_i \log \pi(\theta) + \partial_i k(\theta, \theta')] \quad (16.78)$$

$$[\nabla_f^2 \mathcal{F}_{q_k, \pi}(0)]_{i,j}(\theta', \theta'') = -\mathbb{E}_{\theta \sim q_k} [\partial_{ij}^2 \log \pi(\theta) k(\theta, \theta') k(\theta, \theta'') - \partial_i k(\theta, \theta') \partial_j k(\theta, \theta'')]. \quad (16.79)$$

In order to solve (16.77), Detommaso et al. [417] propose a Galerkin approximation. In particular, let $(\theta_k^i)_{i=1}^n$ be a collection of particles distributed according to q_k . Define the finite-dimensional linear space $\mathcal{H}_n = \text{span} \{k(\theta_k^1, \cdot), \dots, k(\theta_k^n, \cdot)\} \subseteq \mathcal{H}$, and let $\mathcal{H}_n^d = \mathcal{H}_n \times \dots \times \mathcal{H}_n$ denote the corresponding space of vector-valued functions. We can then look for an approximate solution $g = (g_1, \dots, g_d) \in \mathcal{H}_n^d$, viz

$$g_j(\theta) = \sum_{i=1}^n \alpha_i^j k(\theta_k^i, \theta), \quad (16.80)$$

for some real-valued coefficients $(\alpha_i^j)_{i=1:n}^{j=1:d}$ to be determined. In this case, (16.77) reduces to a finite-dimensional linear system, given by

$$\sum_{l=1}^n H_{rl} \alpha_l = b_r, \quad \text{for each } r = 1, \dots, n, \quad (16.81)$$

where $\alpha_l = (\alpha_l^1, \dots, \alpha_l^d)$ is the vector of unknown coefficients, and where

$$[H_{rl}]_{ij} = [\nabla_f^2 \mathcal{F}_{q_k, \pi}(0)]_{ij}(\theta_k^r, \theta_k^l), \quad b_r = -\nabla_f \mathcal{F}_{q_k, \pi}(0)(\theta_k^r). \quad (16.82)$$

By solving (16.81), and substituting the coefficients back into the ansatz (16.80), we obtain an approximation $g = (g_1, \dots, g_d) \in \mathcal{H}_n^d \subseteq \mathcal{H}^d$ for the Newton direction. More precisely, we can only evaluate a Monte Carlo approximation of the Hessian H_{rl} and the gradient b_r using the ensemble $(\theta^i)_{i=1}^n$, which yields an estimate $\hat{g}^n$ of the Newton direction g . The SVN algorithm then reads

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \hat{g}^n(\theta_k^i). \quad (16.83)$$

In practice, computing the solution of the linear system in (16.81) may be prohibitively expensive in high-dimensional problems or when using a large number of particles, and the solution may not be a valid descent direction. Detommaso et al. [417] propose two solutions. The first is to use the inexact Newton-conjugate gradient (NCG) method, which guarantees a descent direction by terminating the conjugate gradient iteration. This approach is also scalable to high dimensions, as it only requires evaluating the matrix-vector product with each H_{rl} . The second is to use a block-diagonal approximation of the second variation, possibly in combination with a Gauss-Newton approximation of the $\nabla^2 \log \pi$, or inexact NCG.

16.5.2 Riemannian SVGD (RSVGD)

Let (Θ, g) be a Riemannian manifold, i.e., a differentiable manifold Θ equipped with a Riemannian metric g , defined as a collection $g = (g_\theta)_{\theta \in \Theta}$ of inner products $g_\theta(\cdot, \cdot) : \mathcal{T}_\theta \Theta \times \mathcal{T}_\theta \Theta \rightarrow \mathbb{R}$ on the tangent spaces $\mathcal{T}_\theta \Theta$, which vary smoothly with θ .

Given a local coordinate chart, the metric g is represented by a $d \times d$ symmetric positive-definite matrix with entries G_{ij} . We can then write the gradient of a function $f : \Theta \rightarrow \mathbb{R}$ and the divergence of a vector field V , as

$$\text{grad} f = \sum_{i,j=1}^d G^{ij} \frac{\partial f}{\partial x_i} \frac{\partial}{\partial x_j} = G^{-1} \nabla f \quad (16.84)$$

$$\text{div}(V) = |G|^{-\frac{1}{2}} \sum_i \frac{\partial}{\partial x_i} \left(|G|^{\frac{1}{2}} V^i \right) = |G|^{-\frac{1}{2}} \nabla \cdot (|G|^{\frac{1}{2}} V), \quad (16.85)$$

where we use G^{ij} to denote the $(i, j)^{\text{th}}$ entry of the inverse matrix G^{-1} , V^i to denote the i^{th} (contravariant) component of V , and where $|G| = \det(G)$.

Following Liu and Zhu [1058], we can now define the *Riemannian-Stein operator* and the *Riemannian-Stein identity* as

$$T_{\pi}^{(G)} \phi := \text{div}_G \phi + \langle \phi, \text{grad} \log \pi \rangle_G, \quad \mathbb{E}_{\pi} [T_{\pi}^{(G)} \phi] = 0, \quad (16.86)$$

where $\langle a, b \rangle_G = a^T G b$, div_G denotes the Riemannian divergence defined above, and π is understood as a density with respect to the Riemannian volume measure. These are the natural generalisations of their (Euclidean) Stein operator and the (Euclidean) Stein identity introduced in Section 16.2.4, cf. (16.12).

Once again, we seek the direction of steepest descent of the KL divergence. Letting $T_{\varepsilon}(\theta) = \theta + \varepsilon \phi(\theta)$ and $q_{\varepsilon} = (T_{\varepsilon})_{\#} q$,³ the derivative of the KL divergence w.r.t. the perturbation magnitude is given by [1058, Theorem 2]

$$\left. \frac{d}{d\varepsilon} \text{D}_{\text{KL}}(q_{\varepsilon} \| \pi) \right|_{\varepsilon=0} = -\mathbb{E}_q [T_{\pi}^{(G)} \phi(\theta)], \quad T_{\pi}^{(G)} \phi := \text{div}_G \phi + \langle \phi, \text{grad} \log \pi \rangle_G. \quad (16.87)$$

Similar to before, the task is now to solve the constrained optimisation problem

$$\phi^* = \arg \min_{\phi \in \mathcal{G}} \left\{ \left. \frac{d}{d\varepsilon} \text{D}_{\text{KL}}(q_{\varepsilon} \| \pi) \right|_{\varepsilon=0} : \|\phi\|_{\mathcal{G}} \leq 1 \right\} = \arg \max_{\phi \in \mathcal{G}} \left\{ \mathbb{E}_q [T_{\pi}^{(G)} \phi(\theta)] : \|\phi\|_{\mathcal{G}} \leq 1 \right\}, \quad (16.88)$$

where $\mathcal{G}$ is a normed space of functions. We also require that ϕ^* is a valid vector field, coordinate invariant, and can be expressed in closed form.

The first and second of these requirements rule out the naive Euclidean choice $\mathcal{G} = \mathcal{H}^d$, the vector-valued RKHS used in the derivation of the standard SVGD update, since its elements are not intrinsically defined vector fields on the manifold. Instead, we will now take

$$\mathcal{G} = \{\text{grad} f : f \in \mathcal{H}\}, \quad (16.89)$$

where $\mathcal{H}$ is the RKHS associated with a positive-definite kernel $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. This gradient is a well-defined object on Θ ; it is guaranteed to be a valid vector field; and it is coordinate invariant [1058, Lemma 3]. Furthermore, it is possible to derive a closed-form solution as [1058, Theorem 4]

$$\phi^* = \text{grad}_{(\cdot)} f^*(\cdot), \quad f^*(\cdot) = \mathbb{E}_{\theta \sim q} [\langle \text{grad}_{\theta} k(\theta, \cdot), \text{grad}_{\theta} \log \pi(\theta) \rangle_{G_{\theta}} + \text{div}_{\theta} (\text{grad}_{\theta} k(\theta, \cdot))], \quad (16.90)$$

where all gradient and divergence operators are understood in the Riemannian sense defined above, with the subscript indicating the variable with respect to which they are taken. The Riemannian SVGD update thus reads

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \text{grad}_{\theta_k^i} \left[\frac{1}{n} \sum_{j=1}^n \left[\langle \text{grad}_{\theta_k^j} k(\theta_k^j, \theta_k^i), \text{grad}_{\theta_k^j} \log \pi(\theta_k^j) \rangle_{G_{\theta_k^j}} + \text{div}_{\theta_k^j} (\text{grad}_{\theta_k^j} k(\theta_k^j, \theta_k^i)) \right] \right]. \quad (16.91)$$

In a fully intrinsic implementation, the Euclidean addition in this update should be replaced by a retraction or exponential-map step on Θ . The term $\text{grad} \log \pi = G^{-1} \nabla \log \pi$ is a preconditioned version of the score function. If one chooses the Riemannian metric to approximate the posterior curvature (e.g., the Fisher information), then this has the effect of *whitening* the score. This can result in accelerated convergence in ill-conditioned models [1058].

³Here, the transport map is expressed in a local coordinate chart; more intrinsically, one would define T_{ε} via the exponential map or a retraction.

16.6 Stochastic and Scalable ParVI

In order to scale ParVI algorithms such as SVGD to large datasets and high-dimensional models, several adaptations are required. These include:

- stochastic approximations of the score function (e.g., mini-batches, variance reduction);
- *noisy* dynamics that retain exploration while preserving descent on the KL divergence;
- *systems-level* techniques (random-feature kernels, GPU/distributed implementations).

We develop these in a way consistent with the functional-gradient view established in Sections 16.2–16.3.

16.6.1 Mini-Batch SVGD

Basic Overview. Let the target posterior be $\pi(\theta) \propto p(\mathcal{D} \mid \theta)p(\theta)$ with $\mathcal{D} = \{(X_n, Y_n)\}_{n=1}^N$. The exact score of the target is given by

$$u_\pi(\theta) = \nabla \log p(\theta) + \sum_{n=1}^N \nabla \log p(Y_n \mid X_n, \theta). \quad (16.92)$$

Given a mini-batch $B_k \subset \{1, \dots, N\}$ of size b , we can define an *unbiased* estimator of the score function as

$$\hat{u}_\pi(\theta; B_k) = \nabla \log p(\theta) + \frac{N}{b} \sum_{n \in B_k} \nabla \log p(Y_n \mid X_n, \theta), \quad (16.93)$$

with $\mathbb{E}_{B_k}[\hat{u}_\pi(\theta; B_k)] = u_\pi(\theta)$. We now replace u_π by $\hat{u}_\pi$ in the SVGD witness function, cf. (16.21), to obtain

$$\hat{\phi}_{q,\pi}^*(\cdot; B_k) = \mathbb{E}_{\theta \sim q} [k(\theta, \cdot) \hat{u}_\pi(\theta; B_k) + \nabla_\theta k(\theta, \cdot)]. \quad (16.94)$$

Substituting this into the population version of the SVGD algorithm, cf. (16.22) or (16.23), we arrive at

$$q_{k+1} = (\hat{T}_{\varepsilon_k}^*)_{\#} q_k, \quad \text{where } \hat{T}_{\varepsilon_k}^*(\theta) = \theta + \varepsilon_k \hat{\phi}_{q_k, \pi}^*(\theta; B_k), \quad (16.95)$$

or equivalently, in terms of a single particle θ_k whose distribution is given by $q_k = \text{Law}(\theta_k)$, the updates

$$\theta_{k+1} = \theta_k + \varepsilon_k \mathbb{E}_{\theta \sim q_k} [k(\theta, \theta_k) \hat{u}_\pi(\theta; B_k) + \nabla_\theta k(\theta, \theta_k)], \quad \theta_0 \sim q_0. \quad (16.96)$$

In practice, of course, one must approximate the expectation w.r.t. the unknown $(q_k)_{k \in \mathbb{N}}$. Similar to before, given $\{\theta_0^i\}_{i=1}^n \stackrel{\text{i.i.d.}}{\sim} q_0$, we will update for $i = 1, \dots, n$:

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \frac{1}{n} \sum_{j=1}^n [k(\theta_k^j, \theta_k^i) \hat{u}_\pi(\theta_k^j; B_k) + \nabla_{\theta_k^j} k(\theta_k^j, \theta_k^i)]. \quad (16.97)$$

This procedure is summarised in Algorithm 21.

Convergence Results. It is natural to ask whether the stochastic transformations defined in (16.95) will still result in a decrease in the KL divergence. That is, for sufficiently small step sizes ε_k , is it true that $D_{\text{KL}}((\hat{T}_{\varepsilon_k}^*)_{\#} q \parallel \pi) \leq D_{\text{KL}}(q \parallel \pi)$?

In (16.33), we reported a descent lemma for SVGD: under suitable conditions, the SVGD algorithm decreases the KL divergence at each iteration. In the stochastic setting, we can establish an analogous result for the *expected descent*. Let $\mu_{q,\pi}$ be the exact witness and $\hat{\mu}_{q,\pi}$ its mini-batch version, so that $\mathbb{E}[\hat{\mu}_{q,\pi}] = \mu_{q,\pi}$. In particular, we have

$$\mathbb{E}[D_{\text{KL}}(q_{k+1} \parallel \pi) \mid q_k] - D_{\text{KL}}(q_k \parallel \pi) \leq -\varepsilon_k \text{KSD}^2(q_k \parallel \pi) + \mathcal{O}(\varepsilon_k^2), \quad (16.98)$$

where the $\mathcal{O}(\varepsilon_k^2)$ term now also depends on the variance of the mini-batch score estimator. Thus, mini-batching preserves the descent of KL in expectation. We note that this statement concerns the population mini-batch dynamics; for the finite-particle algorithm, one must additionally account for the empirical approximation of the witness function. For stronger convergence guarantees (e.g., a.s. convergence), standard Robbins-Monro conditions must be imposed on the step sizes $(\varepsilon_k)_{k \in \mathbb{N}}$.

Algorithm 21: Mini-Batch Stein Variational Gradient Descent (SVGD)**Inputs** : particles $\{\theta_0^i\}_{i=1}^n$; kernel k ; step sizes $\{\varepsilon_k\}$; batch size b **Outputs:** updated particles $\{\theta_K^i\}_{i=1}^n$ **for** $k = 0$ **to** $K - 1$ **do** sample mini-batch B_k of size b compute $\hat{u}_\pi(\theta_k^j; B_k)$ for $j = 1, \dots, n$ **for** $i = 1$ **to** n **do** $\theta_{k+1}^i \leftarrow \theta_k^i + \frac{\varepsilon_k}{n} \sum_{j=1}^n \left[k(\theta_k^j, \theta_k^i) \hat{u}_\pi(\theta_k^j; B_k) + \nabla_{\theta_k^j} k(\theta_k^j, \theta_k^i) \right]$ **return** $\{\theta_K^i\}_{i=1}^n$

Extensions. While computationally convenient, using a mini-batch estimate is not without its drawbacks. For example, while unbiased, the stochastic approximation can have a high variance. There are various approaches to reducing the variance, including:

1. **Control variates.** Define an anchor point (e.g., the maximum a posteriori (MAP) estimate) $\tilde{\theta}$, and use a stochastic variance reduced gradient (SVRG) style estimator:

$$\tilde{u}_\pi(\theta; \tilde{\theta}, B_k) = \hat{u}_\pi(\theta; B_k) - \hat{u}_\pi(\tilde{\theta}; B_k) + u_\pi(\tilde{\theta}). \quad (16.99)$$

This remains unbiased for $u_\pi(\theta)$, and has reduced variance in comparison to $\hat{u}_\pi(\theta)$ [77].

2. **Polyak–Ruppert Averaging.** Use an average of the iterates to dampen stochastic oscillations. This results in a reduced (asymptotic) variance [1338, 1440], and may yield a more stable approximation of the target posterior. Some care is required, however: averaging particle locations is not in general equivalent to improving the posterior approximation, and may be inappropriate for multimodal posteriors.

16.6.2 Stochastic Particle Optimisation Sampling (SPOS)

The stochastic particle optimisation sampling (SPOS) algorithm, introduced in Zhang et al. [1847], augments the Stein variational gradient flow, cf. (16.31) or (16.32), with stochastic gradient Langevin dynamics (SGLD) [509]. In the continuous-time, mean-field limit, this algorithm is described by the McKean-Vlasov PDE

$$\partial_t q_t(\theta) = -\nabla \cdot \left[q_t(\theta) \left(\beta^{-1} u_\pi(\theta) + \int_{\mathbb{R}^d} [k(\theta', \theta) u_\pi(\theta') + \nabla_{\theta'} k(\theta', \theta)] dq_t(\theta') \right) \right] + \beta^{-1} \Delta q_t(\theta), \quad (16.100)$$

or, writing $q_t = \text{Law}(\theta_t)$, the McKean SDE

$$d\theta_t = \left[\beta^{-1} u_\pi(\theta_t) + \int [k(\theta', \theta_t) u_\pi(\theta') + \nabla_{\theta'} k(\theta', \theta_t)] dq_t(\theta') \right] dt + \sqrt{2\beta^{-1}} dw_t, \quad (16.101)$$

where $\beta > 0$ is a temperature parameter.

In the limit as the temperature $\beta \rightarrow \infty$, the Langevin drift and diffusion vanish, and (16.101) formally reduces to the continuous-time, mean-field SVGD dynamics, cf. (16.32); meanwhile, if the particle interaction term is removed or becomes negligible (e.g., in the limit of vanishing kernel bandwidth, in which the repulsion between distinct particles vanishes), SPOS reduces to a tempered variant of SGLD.

In practice, of course, we will approximate (16.101) using (i) a finite collection of particles and (ii) an appropriate time discretisation (e.g., Euler Maruyama). We can also use a stochastic (e.g., mini-batch) estimate for the score function. This yields the final SPOS algorithm:

$$\theta_{k+1}^i = \theta_k^i + \varepsilon_k \left[\beta^{-1} \hat{u}_\pi(\theta_k^i; B_k) + \frac{1}{n} \sum_{j=1}^n \left(k(\theta_k^j, \theta_k^i) \hat{u}_\pi(\theta_k^j; B_k) + \nabla_{\theta_k^j} k(\theta_k^j, \theta_k^i) \right) \right] + \sqrt{2\varepsilon_k \beta^{-1}} \xi_k^i, \quad (16.102)$$

where $\xi_k^i \sim \mathcal{N}(0, I)$ are a sequence of i.i.d. standard normal random variables.

Convergence Results. Under appropriate assumptions (e.g., log-concavity, bounded initial moments, constant or decreasing step sizes), it is possible to obtain *non-asymptotic* convergence rates for SPOS in the 1-Wasserstein metric W_1 [1847, Theorem 9, Theorem 10]. In practice, the noise term improves exploration and mitigates spurious fixed points of the deterministic Stein flow.

16.6.3 Random-feature kernels and sub-quadratic interactions

In each iteration, SVGD requires us to compute $\{k(\theta^i, \theta^j)\}_{i,j=1}^n$. Its cost is thus $\mathcal{O}(n^2)$ per iteration. If a very large number of particles is required, there are several options for reducing the computational cost:

- **Random Fourier Features.** For shift-invariant kernels $k(\theta, \theta') = \kappa(\theta - \theta')$, we can use the approximation $k(\theta, \theta') \approx \varphi(\theta)^\top \varphi(\theta')$ with m random Fourier features $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^m$. In this case,

$$\phi_{q,\pi}(\cdot) \approx \underbrace{\mathbb{E}_{\theta \sim q} [\varphi(\theta) u_\pi(\theta)^\top]^\top}_{\in \mathbb{R}^{d \times m}} \varphi(\cdot) + \underbrace{\mathbb{E}_{\theta \sim q} [J_\varphi(\theta)]^\top}_{\in \mathbb{R}^{d \times m}} \varphi(\cdot), \quad (16.103)$$

where $J_\varphi(\theta) \in \mathbb{R}^{m \times d}$ denotes the Jacobian of φ at θ ; in mini-batch settings, u_π can be replaced by the stochastic estimate $\hat{u}_\pi$. This reduces the particle-interaction cost from $\mathcal{O}(n^2)$ to $\mathcal{O}(nm)$, up to the cost of evaluating the features and their Jacobians [1064].

- **Structured features.** The use of structured features (e.g., orthogonal random features) can further improve the computational cost.

16.6.4 GPU and distributed implementations

In the single-node GPU setting, both the kernel matrix $\mathbf{K} = [k(\theta^i, \theta^j)]_{ij}$ and score vector $\mathbf{u}_\pi = [\hat{u}_\pi(\theta^i)]_i$ are amenable to GPU parallelism. In particular, JIT-compiled implementations (e.g., PyTorch, JAX) fuse the two terms in (16.97) into batched tensor kernels.

For multi-node GPUs, with M devices, each holding roughly n/M particles and a shard of data, a typical iteration would be:

1. compute local scores on local batches;
2. *all-reduce* particle features (or sufficient statistics for (16.103));
3. update local particles.

In the last step, there are several ways to avoid global $\mathcal{O}(n^2)$ communication. These include using a random-feature factorisation (16.103) and *all-reduce* of the m -dimensional summaries; block-sparse kernels (nearest-neighbour graphs); or inducing-particle approximations (analogous to sparse GPs).

16.7 Applications in Bayesian Deep Learning

ParVI methods have been applied to a wide range of problems in BDL, offering scalable alternatives to exact Bayesian inference in high-dimensional, non-convex, and multimodal posterior landscapes.

16.7.1 Bayesian Neural Networks

Bayesian neural networks (BNNs) define a posterior over network parameters θ given data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$:

$$\pi(\theta) \propto p(\mathcal{D} \mid \theta) p(\theta), \quad (16.104)$$

where $p(\mathcal{D} \mid \theta)$ is the likelihood induced by the network outputs $f(\cdot, \theta)$, and $p(\theta)$ is the prior over the network parameters.

Posterior Prediction. Using a BNN, predictions for new inputs x_* are obtained by computing the *posterior predictive distribution*, defined as

$$p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, \theta) \pi(\theta) d\theta. \quad (16.105)$$

In modern architectures, exact computation of the posterior is infeasible, and the integral in (16.105) must be approximated. ParVI methods such as SVGD approximate π using an ensemble of particles $\{\theta^i\}_{i=1}^n$, and can be readily applied in this context. In particular, these particles can be used to approximate the posterior predictive distribution as

$$p(y_* \mid x_*, \mathcal{D}) \approx \frac{1}{n} \sum_{i=1}^n p(y_* \mid x_*, \theta^i). \quad (16.106)$$

SVGD and related Stein ensemble methods have shown competitive empirical performance in BNNs, for both regression and classification tasks. For example, on image classification benchmarks such as CIFAR-10 and MNIST, SVGD with $n \approx 20$ particles has been reported to outperform competing methods (e.g., deep ensembles) in terms of predictive accuracy, functional diversity, and uncertainty quantification [368, 1064, 1704], although such gains are problem- and architecture-dependent.

We highlight, in particular, the work of D’Angelo et al. [368], which unifies several variants of SVGD applicable to BNNs under the title of *Stein variational neural network ensembles*. Their framework includes the original SVGD algorithm – which performs inference directly in the weight space, and uses a kernel defined in the weight space – as well as the standard deep ensemble, which can be viewed as SVGD with a kernel defined by an indicator function.

It also includes several novel variants of SVGD, designed specifically for BNNs, which are derived via careful choices of the kernel function. These include *function-kernel weight SVGD*, in which the kernel is defined in the output space of the neural network:

$$k_f(\theta, \theta') = \exp \left(-\frac{1}{h} \mathbb{E}_{x \sim p(x)} [\|f(x; \theta) - f(x; \theta')\|^2] \right), \quad (16.107)$$

where $h > 0$ is a bandwidth parameter, $p(x)$ denotes the marginal distribution of the inputs, and $f(x; \theta)$ denotes the output of the neural network with parameters θ , given inputs x . In practice, this kernel is approximated using a Monte Carlo estimate over a mini-batch of inputs, typically sampled from the empirical training distribution. This choice of kernel, as shown in D’Angelo et al. [368], encourages functional diversity of the particles and avoids functional posterior collapse. This results in improved calibration, a greater resemblance to the true posterior distribution, and enhanced out-of-distribution (OOD) detection.

Out-of-Distribution Detection. The problem of out-of-distribution detection can be described as follows: given a test point x_* , decide whether x_* is from the same distribution as the training data, or a different distribution. Given a test point x_* , the predictive entropy

$$\mathcal{H}[p(y_* | x_*, \mathcal{D})] = - \sum_{y_*} p(y_* | x_*, \mathcal{D}) \log p(y_* | x_*, \mathcal{D}) \quad (16.108)$$

can be used as an out-of-distribution score. In particular, smaller values of the entropy correspond to confident predictions (i.e., likely to be in distribution); larger values correspond to uncertain predictions (i.e., possible out of distribution). In ParVI methods such as SVGD (and variants), disagreement among particles in predictive space directly increases the entropy for test inputs far from the training data, thereby providing a useful OOD signal; D’Angelo et al. [368] report improved OOD detection relative to methods such as deep ensembles, although relative performance is problem-dependent. See also Chapter 44 for a complementary treatment of predictive-uncertainty-based rejection.

Active Learning. In (Bayesian) active learning, the objective is to select the most informative data points to query in order to improve the performance of the model (e.g., the BNN). In Bayesian models, a popular acquisition function is the decrease in the expected posterior entropy, viz

$$\text{MI}(y_*, \theta | x_*, \mathcal{D}) = \mathcal{H}[p(y_* | x_*, \mathcal{D})] - \mathbb{E}_{\theta \sim \pi(\theta)} [\mathcal{H}(p(y_* | x_*, \theta))]. \quad (16.109)$$

This is known as *Bayesian active learning by disagreement* (BALD); see Chapter 53 for a detailed treatment of Bayesian active learning. Using ParVI methods (e.g., SVGD), it is straightforward to approximate this acquisition function, replacing the posterior π by its particle-based approximation.

16.7.2 Large-Scale Models and Bayesian Fine-Tuning

For very large models (e.g., large language models or vision foundation models), a fully Bayesian treatment over all parameters is usually computationally prohibitive, and can be difficult to justify relative to cheaper uncertainty quantification strategies [277, 308]. The large number of parameters renders even approximate posterior inference infeasible, and the vast size of the (pre)training dataset often leads to relatively low posterior uncertainty in many parameters.

On the other hand, during fine-tuning, the number of learnable parameters is much smaller, and we typically have a much smaller dataset, implying significantly more uncertainty. There is, therefore, a stronger justification for the additional expense associated with the use of Bayesian methods.

Let θ denote the parameters of a large pre-trained model, and ϕ denote the parameters (e.g., of a small adapter, of the last layer) that we wish to fine-tune based on downstream data $\mathcal{D}$. We would like to compute the posterior over the fine-tuning parameters, viz

$$\pi(\phi) := p(\phi|\mathcal{D}, \theta) \propto p(\mathcal{D} | \theta, \phi)p(\phi), \quad (16.110)$$

where, similar to before, $p(\mathcal{D} | \theta, \phi)$ denotes the likelihood induced by the model outputs, and $p(\phi)$ denotes a prior over the fine-tuning parameters.

Since exact inference in $\pi(\phi)$ is generally intractable, we can approximate it using ParVI. In particular, using a ParVI algorithm, one obtains n particles $\{\phi^i\}_{i=1}^n$, each representing the parameters of a slightly different fine-tuned model. This yields an approximate posterior predictive distribution given by

$$p(y_* | x_*, \mathcal{D}) \approx \frac{1}{n} \sum_{i=1}^n p(y_* | x_*, \phi^i, \theta), \quad (16.111)$$

which can quantify (epistemic) uncertainty in low-data regimes [432, 1286]. ParVI-based fine-tuning is compatible with parameter-efficient strategies (e.g., low-rank adaptation (LoRA), bias-only tuning), keeping dimensionality manageable; see Chapters 39 and 43 for detailed treatments of last-layer inference and Bayesian approaches for large language models, respectively. Extensions to continual learning are also possible: as new datasets $\mathcal{D}_1, \mathcal{D}_2, \dots$ arrive, the particle set can be propagated via sequential ParVI updates (Chapter 18), analogous to particle filtering.

16.8 Theoretical Guarantees and Open Problems*

Despite impressive empirical performance, ParVI algorithms such as SVGD still lack the kind of sharp, dimension-aware theory that is now standard for diffusion-based samplers (e.g., Langevin Monte Carlo, Hamiltonian Monte Carlo). This section summarises existing theoretical results for SVGD, including (i) the continuous-time, infinite-particle dynamics; (ii) the discrete-time, infinite-particle dynamics; (iii) the continuous-time, finite-particle dynamics; and (iv) the discrete-time, finite-particle dynamics. We conclude this section by outlining some open problems and interesting directions for future work.

16.8.1 Continuous Time, Infinite-Particle SVGD

In this section, we review convergence results for the continuous-time, infinite-particle SVGD dynamics, viz

$$\frac{\partial q_t(\theta)}{\partial t} = -\nabla_\theta \cdot (q_t(\theta)\phi_{q_t, \pi}^*(\theta)), \quad \phi_{q_t, \pi}^*(\cdot) = \mathbb{E}_{\theta \sim q_t} [k(\theta, \cdot)u_\pi(\theta) + \nabla_\theta k(\theta, \cdot)]. \quad (16.112)$$

Well-Posedness. Under smoothness and growth conditions on the target π and the kernel k , this PDE is well posed (i.e., global existence, uniqueness, regularity) [1065]. Although π is a stationary solution of the PDE, establishing uniqueness of the invariant solution, and convergence of q_t to π , generally requires further assumptions: e.g., that a vanishing KSD characterises π (i.e., the KSD is convergence-determining for the chosen kernel), or that the target satisfies an appropriate functional inequality (see below).

Dissipation Identity. Using calculus on the space of probability measures, the dissipation of the KL divergence along the continuous-time, mean-field SVGD dynamics is given by [918, Proposition 1]

$$\frac{d}{dt} D_{\text{KL}}(q_t \| \pi) = -\mathcal{I}_{\text{Stein}}(q_t \| \pi), \quad (16.113)$$

where $\mathcal{I}_{\text{Stein}}(q \| \pi)$ denotes the *Stein-Fisher information* of q relative to π , which coincides with the squared kernel Stein discrepancy (KSD) [462]

$$\mathcal{I}_{\text{Stein}}(q \| \pi) := \text{KSD}^2(q \| \pi). \quad (16.114)$$

Thus, in particular, KL is a Lyapunov functional and the (squared) KSD is the dissipation rate. Under additional assumptions (e.g., bounded first moment, bounded kernel), (16.113) implies that $\mathcal{I}_{\text{Stein}}(q_t \| \pi) \rightarrow 0$ as $t \rightarrow \infty$ [918, Proposition 2]. Thus, for appropriate choices of the kernel, and given suitable tightness or moment conditions, this further implies that $q_t \Rightarrow \pi$ [652]. Another consequence of (16.113) is that [918, Proposition 3]

$$\min_{0 \leq s \leq t} \text{KSD}^2(q_s \| \pi) \leq \frac{1}{t} \int_0^t \text{KSD}^2(q_s \| \pi) ds \leq \frac{D_{\text{KL}}(q_0 \| \pi)}{t}. \quad (16.115)$$

Convergence Rates. Under the assumption that the target π satisfies an appropriate functional inequality, it is possible to quantify how fast the flow converges to π . For example, if the target satisfies a *Stein log-Sobolev inequality* (LSI) with constant λ , viz

$$D_{\text{KL}}(q\|\pi) \leq \frac{1}{2\lambda} \mathcal{I}_{\text{Stein}}(q\|\pi), \quad (16.116)$$

then the continuous-time, mean-field SVGD dynamics converge exponentially fast to the target distribution [918, Proposition 4]

$$D_{\text{KL}}(q_t\|\pi) \leq e^{-2\lambda t} D_{\text{KL}}(q_0\|\pi). \quad (16.117)$$

16.8.2 Discrete-Time, Infinite-Particle SVGD

We next consider the forward-Euler discretisation of the continuous-time, infinite-particle dynamics in (16.112), viz

$$q_{k+1} = (\text{id} + \varepsilon_k \phi_{q_k, \pi}^*)_{\#} q_k, \quad \phi_{q_k, \pi}^*(\cdot) = \mathbb{E}_{\theta \sim q_k} [k(\theta, \cdot) u_{\pi}(\theta) + \nabla_{\theta} k(\theta, \cdot)], \quad (16.118)$$

where $(\varepsilon_k)_{k \in \mathbb{N}}$ is a positive, non-increasing step-size schedule. This corresponds to the *population limit* or *mean-field limit* of the SVGD algorithm.

A First Descent Lemma. The original work of Liu [1063] provided the first theoretical guarantees for SVGD in discrete time at the population level. In particular, assuming boundedness of the KSD and the kernel, Liu [1063, Theorem 3.3] established a *descent lemma*, guaranteeing that for sufficiently small step sizes the KL decreased at each iteration. That is,

$$D_{\text{KL}}(q_{k+1}\|\pi) - D_{\text{KL}}(q_k\|\pi) \leq -\varepsilon_k \text{KSD}^2(q_k\|\pi) + \varepsilon_k^2 R \cdot \text{KSD}^2(q_k\|\pi), \quad (16.119)$$

where $0 < R < \infty$ is a positive constant depending on the kernel and the score, namely, $R = \sup_x [\frac{1}{2} \|\nabla \log p\|_{\text{Lip}} k(x, x) + 2\nabla_{xx} k(x, x)]$.

Initial Non-Asymptotic Convergence Rates. Subsequently, Korba et al. [918] obtained a similar descent lemma and established the first non-asymptotic convergence rate for the population version of the SVGD algorithm. To be specific, assuming the kernel and its first derivative are bounded by $0 < B < \infty$; the target is L -smooth, so that the Hessian of the target is bounded by $0 < L < \infty$; and the KSD along the path of the algorithm iterates is bounded by $0 < C < \infty$, it was established in Korba et al. [918, Proposition 5] that

$$D_{\text{KL}}(q_{k+1}\|\pi) - D_{\text{KL}}(q_k\|\pi) \leq -\varepsilon_k \text{KSD}^2(q_k\|\pi) + \varepsilon_k^2 \frac{(\alpha^2 + L)B^2}{2} \cdot \text{KSD}^2(q_k\|\pi), \quad (16.120)$$

for any $\alpha > 1$, provided the step size $\varepsilon_k := \varepsilon \leq \frac{\alpha-1}{\alpha B C^{\frac{1}{2}}}$. Although this descent lemma is rather similar to the one previously obtained in Liu [1063, Theorem 3.3], it was based on rather different proof techniques, namely, differential calculus on the Wasserstein space.

Following this result, Korba et al. [918] also obtained a $\mathcal{O}(\frac{1}{K})$ convergence rate for the arithmetic mean of the squared KSD between the iterates q_k and π . In particular, writing $c_{\varepsilon} = \varepsilon(1 - \varepsilon \frac{(\alpha^2 + L)B^2}{2})$, and assuming a similar condition on the step size, the authors showed that [918, Corollary 6]

$$\min_{k=0, \dots, K-1} \text{KSD}^2(q_k\|\pi) \leq \frac{1}{K} \sum_{k=0}^{K-1} \text{KSD}^2(q_k\|\pi) \leq \frac{D_{\text{KL}}(q_0\|\pi)}{c_{\varepsilon} K}. \quad (16.121)$$

Complexity Bounds under Talagrand's T_1 Inequality. Following these results, Salim et al. [1451] obtained the first dimension-explicit iteration complexity bounds for SVGD, assuming the target π satisfied Talagrand's T_1 inequality. Notably, unlike in previous works, their results did not require the assumption that the KSD was uniformly bounded along the algorithm iterates, an assumption which cannot be verified prior to running the algorithm.

To be more specific, Salim et al. [1451] first established a descent lemma [1451, Theorem 3.2] and a non-asymptotic convergence rate for the average of the squared KSD [1451, Corollary 4.2]. Their results are essentially identical to the corresponding results in Korba et al. [918], except that now the dependence on the kernel bound is linear rather than quadratic, and the bound on the step size does not require a bound on the KSD. Finally, the authors provided the first dimension-dependent complexity result for SVGD, showing that, provided the number of iterations satisfies [1451, Corollary 4.3]

$$K = \tilde{\Omega} \left(\frac{L d^{\frac{3}{2}}}{\lambda^{\frac{1}{2}} \varepsilon} \right), \quad (16.122)$$

one obtains $\text{KSD}^2(\bar{q}_K \parallel \pi) \leq \varepsilon$, where λ denotes the constant in Talagrand's T_1 inequality; and $\bar{q}_K = \frac{1}{K} \sum_{k=0}^{K-1} q_k$ denotes the average of the iterates.

Convergence under Weaker Smoothness Assumptions. Most recently, Sun et al. [1575] established the convergence of SVGD (i.e., a descent lemma, convergence rate, iteration complexity) under weaker assumptions. In particular, their results required that (a) the target satisfies a T_p inequality with constant λ or a generalised (T_p, J) inequality; and (b) the target potential is (L_0, L_1) -smooth, i.e., there exist constants $L_0, L_1 > 0$ such that $\|\nabla^2 V(\theta)\|_{\text{op}} \leq L_0 + L_1 \|\nabla V(\theta)\|$ for all $\theta \in \mathbb{R}^d$. This significantly extended previous results, all of which had assumed L -smoothness of the potential (i.e., global Lipschitz continuity of the score, globally bounded Hessian), thereby excluding a large class of potential functions (e.g., polynomials of degree greater than two).

16.8.3 Finite-Particle SVGD

Finally, we review convergence results for the finite-particle system. We consider both the continuous-time dynamics, viz

$$\dot{\theta}_t^i = \frac{1}{n} \sum_{j=1}^n \left[k(\theta_t^j, \theta_t^i) u_\pi(\theta_t^j) + \nabla_{\theta_t^i} k(\theta_t^j, \theta_t^i) \right], \quad i = 1, \dots, n, \quad (16.123)$$

and its discrete-time counterpart, i.e., the SVGD algorithm itself, viz

$$\theta_{t+1}^i = \theta_t^i + \varepsilon_t \frac{1}{n} \sum_{j=1}^n \left[k(\theta_t^j, \theta_t^i) u_\pi(\theta_t^j) + \nabla_{\theta_t^i} k(\theta_t^j, \theta_t^i) \right], \quad i = 1, \dots, n, \quad (16.124)$$

where, in a slight abuse of notation, we now use t to index both the time in the continuous-time dynamics; and the iteration in the discrete-time algorithm.

General Strategies For Establishing Convergence. Broadly speaking, establishing convergence of the finite-particle system can proceed via two possible routes. Similar to before, let $q_t^n = \frac{1}{n} \sum_{j=1}^n \delta_{\theta_t^j}$ denote the empirical distribution of the interacting particle system. Then we can either:

- (i) Show that $\lim_{t \rightarrow \infty} [\lim_{n \rightarrow \infty} q_t^n] = \lim_{t \rightarrow \infty} q_t = \pi$. That is:
 - (i.a) **Propagation of Chaos.** First establish that the empirical distribution q_t^n converges to the mean-field distribution q_t as $n \rightarrow \infty$, for any finite time $t \in [0, \infty)$.
 - (i.b) **Infinite-Particle Convergence to Equilibrium.** Then establish that the mean-field distribution q_t converges to the target distribution π as $t \rightarrow \infty$. This was the subject of Sections 16.8.1 and 16.8.2.
- (ii) Show that $\lim_{n \rightarrow \infty} [\lim_{t \rightarrow \infty} q_t^n] = \lim_{n \rightarrow \infty} \pi^n = \pi$. That is:
 - (ii.a) **Finite-Particle Convergence to Equilibrium.** First establish that the empirical distribution q_t^n converges to a (unique) equilibrium, say π^n , as $t \rightarrow \infty$.
 - (ii.b) **Uniform-In-Time Propagation of Chaos.** Then establish that the empirical equilibrium distribution π^n converges to the target π as $n \rightarrow \infty$.

We refer to Banerjee et al. [90], Liu et al. [1067] for a visual illustration of these two approaches. In the subsequent discussion, we will focus on (i), this being the prevailing approach in the literature. Regarding (ii), we mention briefly the recent work of Liu et al. [1067], which analysed this approach for Gaussian SVGD, wherein both the target distribution and the initial distribution are Gaussian, and the kernel is bilinear. In this case, the mean-field distribution remains Gaussian for all times [e.g., 1067, Theorem 3.1] and it is possible to establish quantitative uniform-in-time convergence of the finite particle system to its mean-field limit [e.g., 1067, Theorem 3.7]. Most recently, Balasubramanian et al. [79] have established uniform-in-time propagation-of-chaos bounds for SVGD, obtaining rates with logarithmic dependence on the number of particles in general settings, and parametric $\mathcal{O}(n^{-1/2})$ rates, uniform in time, for Gaussian targets with matrix-valued kernels. Karimi Jaghargh et al. [871], Priser et al. [1346] also study finite-particle asymptotics.

Propagation of Chaos. Several different papers have established the time-dependent convergence of the finite-particle system to its mean-field limit, in both continuous time [241, 1085] and discrete time [653, 918, 1063], under various assumptions on the kernel (e.g., continuous, Lipschitz, smooth) and the potential (e.g., uniformly bounded). For example, regarding the discrete-time algorithm, Korba et al. [918] showed that, for any fixed finite horizon T , and for all iterations t corresponding to times up to T ,

$$\mathbb{E} [W_2^2(q_t, q_t^n)] \leq \frac{1}{2} \left(\frac{1}{\sqrt{n}} \sqrt{\text{Var}(q_0)} e^{LT} \right) (e^{2LT} - 1), \quad (16.125)$$

where L is a constant depending on the kernel k and the target π ; see [918, Section 11.4] for details.

Unified Convergence Analyses. By combining convergence of the finite-particle distribution q_t^n to the mean-field distribution q_t as $n \rightarrow \infty$ (see above), and convergence of the mean-field distribution q_t to the target distribution π as $t \rightarrow \infty$, two groups have now obtained a unified convergence analysis for the SVGD algorithm [90, 1512].

The first such result was established in Shi and Mackey [1512], who combined improved rates for $q_t^n \rightarrow q_t$ [1512, Theorem 2], with existing rates for $q_t \rightarrow \pi$ [1451, Theorem 3.2], to obtain a bound on $\min_{0 \leq s \leq t} \text{KSD}(q_s^n \parallel \pi)$. However, due to the doubly exponential dependence on time of their bound between the finite-particle distribution and the mean-field distribution (via the sum of the step sizes), the obtained rate is rather slow, namely, of order $1/\sqrt{\log \log n}$.

More recently, Banerjee et al. [90] have obtained significantly improved finite-particle convergence rates for SVGD, in both KSD and Wasserstein-2 metrics. In particular, the authors improve the dependence on the number of particles from the $1/\sqrt{\log \log n}$ rate of Shi and Mackey [1512] to $1/\sqrt{n}$ (a double-exponential improvement), in both continuous and discrete time. Under mild assumptions on the kernel and potential, these bounds also grow polynomially in the dimension d .

16.8.4 Curse-of-Dimensionality

The curse of dimensionality largely affects ParVI algorithms such as SVGD through the geometry of the RKHS.

Even for simple Gaussian targets, SVGD with standard (e.g., radial basis function (RBF), IMQ) kernels can systematically underestimate the posterior variance in high dimensions [e.g., 68, 1882]. This is known as the *variance collapse phenomenon*. Indeed, for RBF kernels, the magnitude of the repulsive term in the Stein update equation, cf. (16.35), can be upper bounded by a quantity that decreases with d [e.g., 1882, Proposition 1]. Thus, the particles tend to exhibit mode collapse.

In Ba et al. [68], the authors undertake a detailed analysis of this phenomenon. They argue that, in high dimensions, the “driving force” (i.e., the kernel-weighted score function) has a higher variance than the repulsion [68, Proposition 2]. This, coupled with the deterministic nature of the SVGD updates (i.e., no noise or resampling), leads to systematic underestimation of the variance [68, Section 4.2]. The authors then consider the limit as $n, d \rightarrow \infty$, with $\frac{d}{n} \rightarrow \gamma$ for some $\gamma > 1$. For an isotropic Gaussian target, it is possible to explicitly characterise the behaviour of the SVGD variance as a function of γ , for two common choices of the kernel (i.e., RBF, IMQ). In both cases, the learned variance decreases as γ increases, suggesting the number of particles should grow linearly with d in order to reliably estimate the dimension-averaged variance.

The variance collapse phenomenon has motivated various methodological developments. Structural approaches reduce the effective dimensionality, including message-passing SVGD [1882], projected SVGD [270], sliced SVGD [646], Grassmann SVGD [1068], and function-space SVGD [368]. Meanwhile, geometric approaches employ pre-conditioned or matrix-valued kernels to adaptively whiten the score [1704]. Finally, stochastic variants of SVGD, such as SPOS [1847], inject noise into the dynamics to counteract deterministic collapse.

16.8.5 Open problems

We conclude this section by highlighting a number of open problems.

Kernel Design in High Dimensions. In most applications, practitioners resort to the use of an RBF kernel, with the bandwidth chosen adaptively according to a median heuristic [e.g., 1064]. In very high dimensions, this choice is often ineffective. Open questions include the design of principled, problem-dependent kernels (e.g. function-space, neural tangent kernel (NTK)-like, or Fisher-preconditioned kernels) and adaptive, iteration-dependent kernels; and guarantees that connect kernel choice to *dimension-robust* rates.

Finite-Particle Degeneracy. We currently lack general (e.g., algorithm-independent) conditions that preclude particle collapse for a broad class of targets. Practical heuristics (e.g., an adaptive bandwidth, tempering, occasional noise injection, MCMC rejuvenation) can help, but remain outside of the current theory. Meanwhile, obtaining improved non-asymptotic, finite-particle convergence rates under weaker assumptions remains an ongoing and active topic of research.

Beyond the KSD. Typical convergence rates – including infinite-particle and finite-particle – are stated in terms of the KSD [e.g., 90, 918, 1512, 1575]. In reality, however, practitioners may be interested in other metrics such as the predictive risk, coverage, etc. An open question is when (and at what rate) does decreasing the KSD ensure calibrated predictive distributions? Currently, there exist only partial results assuming additional structure on the target (e.g., Gaussian) or the kernel (e.g., composite, bilinear).

Theory for SVGD with Stochastic Gradients. Typical applications of ParVI in BDL (see Section 16.7.1) necessitate the use of stochastic gradients and/or the injection of additional noise. We currently lack a complete theory for mini-batch SVGD (e.g., with variance reduction) and for noisy variants of SVGD (e.g., SPOS).

Differential Privacy. Differential privacy for ParVI algorithms such as SVGD remains largely unexplored. While results exist for other sampling algorithms such as stochastic gradient MCMC (Chapter 2) [e.g., 1021], the questions of how particle interactions affect privacy loss and how to calibrate ParVI algorithms for differential privacy, without destroying accuracy or causing particle collapse, have not yet been studied in detail.

16.9 Summary

In this chapter, we introduced particle-based variational inference, a family of methods which combine the non-parametric flexibility of MCMC with the optimisation-based efficiency of VI. Starting from SVGD, we showed how ParVI methods can be understood as time discretisations and particle approximations of gradient flows on the space of probability measures, and described a number of second-order, Riemannian, and stochastic variants. We then surveyed applications of these methods in Bayesian deep learning, ranging from Bayesian neural networks to the Bayesian fine-tuning of large-scale models, before reviewing existing theoretical guarantees. ParVI remains a rapidly developing field, and the open problems outlined in the previous section – including kernel design in high dimensions, finite-particle degeneracy, and theory for stochastic variants – are likely to remain active topics of research in the coming years.

Chapter 17

Variational inference in function space

17.1 Introduction

Bayesian neural networks (BNNs) were initially envisioned as a framework that could unify the representational power of deep learning with the principled uncertainty quantification of Bayesian inference. Yet, for particular approximation families and experimental settings, standard weight-space posterior approximations can underperform in terms of predictive accuracy or uncertainty calibration [542, 826]. A key difficulty of the standard weight-space variational formulation lies in the nature of inference in weight space: the map from network parameters to functions is highly nonlinear and many-to-one, leading to complex, multimodal posterior landscapes. This makes it hard both to approximate posteriors in this formulation and to interpret or encode prior knowledge about the data-generating process through a weight-space prior.

These challenges motivate a shift of perspective from parameter-space to function-space variational inference (FSVI). At the theoretical level, rather than performing inference over neural-network weights, one works with probability measures over functions, objects that describe input–output behaviour and more naturally capture inductive biases about the learning problem. In this setting, the prior is defined over functions (for instance, as a Gaussian process), and the likelihood operates on function evaluations. FSVI seeks an approximate posterior measure by minimising its divergence from the posterior measure. Practical methods, however, always must access these measures through finite-dimensional representations either through parameters or through finite-dimensional marginals since it is infeasible to manipulate an infinite-dimensional law in its entirety.

This change in perspective has important implications for both the mathematical and computational structure of variational inference. First, it requires measure-theoretic formulations of Bayes’ rule and Kullback–Leibler (KL) divergence in infinite-dimensional spaces, where no analogue of Lebesgue measure exists. Second, it exposes limitations of KL-based objectives: they may be intractable to evaluate directly and can be infinite in many situations, e.g., when a finite-dimensional parameterised neural network represents a variational function law and the prior is a non-degenerate Gaussian process. These difficulties motivate a range of KL approximations and generalised objectives based on alternative discrepancies.

This chapter develops the function-space perspective. We formalise Bayesian inference on a measurable function space, derive the associated variational objectives, and show how finite-dimensional approximations make them usable and motivate new methods. Throughout, we emphasise the interplay between function-space theory and its practical realisation in neural networks and Gaussian processes.

17.2 Function-Space Bayes Theorem

The key difficulty when working with function spaces is that there is no analogue of a translation-invariant, σ -finite Lebesgue measure whenever the corresponding space is infinite-dimensional. Consequently, Bayes’ rule cannot be written as “posterior density $\propto$ prior density $\times$ likelihood” relative to a base Lebesgue measure. Instead, the correct formulation is *measure-theoretic*: the posterior is defined via the Radon–Nikodym derivative of the posterior measure with respect to the prior measure. For a broader perspective on the measure-theoretic formulation of variational inference, see [1149] and the references therein.

17.2.1 Setting and Notation

Let $\mathcal{X}$ be a measurable input domain and let $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$ be a space of real-valued functions equipped with a σ -algebra $\mathfrak{F}$. We use $m \in \mathbb{N}$ for the size of an arbitrary finite evaluation tuple. For $\mathbf{x} = (x_1, \dots, x_m) \in \mathcal{X}^m$, define the **evaluation map**

$$\text{eval}_{\mathbf{x}} : \mathcal{F} \rightarrow \mathbb{R}^m, \quad \text{eval}_{\mathbf{x}}(f) := [f(x_1), \dots, f(x_m)]^\top.$$

Two examples are: (i) $\mathcal{F} = \mathbb{R}^{\mathcal{X}}$ with the cylinder, or product, σ -algebra, namely the smallest σ -algebra making every finite evaluation map measurable; and (ii) a separable Banach or Hilbert space of functions with its Borel σ -algebra. Reproducing kernel Hilbert spaces (RKHSs), introduced in Chapter 23, are convenient examples because their point evaluations are continuous linear functionals and hence measurable.

We write $\mathbf{p}$ and $\mathbf{q}$ for probability measures on $(\mathcal{F}, \mathfrak{F})$ (e.g., a prior measure $\mathbf{p}$ and a variational approximation $\mathbf{q}$). The pushforward finite-dimensional marginal of $\mathbf{p}$ is $\mathbf{p}_{\mathbf{x}} := (\text{eval}_{\mathbf{x}})_{\#}\mathbf{p}$.

Let $(\mathcal{Y}, \mathfrak{Y})$ denote the data space, equipped with the corresponding σ -algebra $\mathfrak{Y}$. A **likelihood functional** is a map

$$L(\cdot | f) : \mathcal{Y} \rightarrow [0, \infty], \quad \text{such that } (f, y) \mapsto L(y | f) \text{ is } \mathfrak{F} \otimes \mathfrak{Y}\text{-measurable.}$$

For example, if the data space is $\mathcal{Y} = \mathbb{R}^n$, then $L(y | f)$ may be a density $p(y | f)$ with respect to Lebesgue measure on $\mathbb{R}^n$, parameterised by $f \in \mathcal{F}$.

In supervised learning, let n denote the number of observations, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, and write $\mathbf{x}_{\text{tr}} = (x_1, \dots, x_n) \in \mathcal{X}^n$ and $y = (y_1, \dots, y_n) \in \mathcal{Y}$. A common factorised likelihood is

$$L(y | f) = \prod_{i=1}^n p(y_i | f(x_i)) = \tilde{L}(y | \text{eval}_{\mathbf{x}_{\text{tr}}}(f)),$$

so the likelihood depends on f only through its values at the training inputs. Other settings are possible. For example, in Cox process models, f parameterises a Poisson-process intensity and the likelihood depends on integrals involving f ; see, e.g., [1148].

17.2.2 Function-Space Bayes Theorem

Fix an observation $y \in \mathcal{Y}$ and let $\mathbf{p}$ denote the prior probability measure. Then the *evidence* (marginal likelihood) is the Lebesgue integral

$$Z(y) := \int_{\mathcal{F}} L(y | f) d\mathbf{p} \in [0, \infty].$$

We will assume throughout this Chapter that $0 < Z(y) < \infty$. Define a set function on $\mathfrak{F}$ by

$$\mathbf{p}(A | y) := \frac{1}{Z(y)} \int_A L(y | f) d\mathbf{p}, \quad A \in \mathfrak{F}.$$

Then $\mathbf{p}(\cdot | y)$ is a probability measure on $(\mathcal{F}, \mathfrak{F})$, it is absolutely continuous with respect to $\mathbf{p}$ (denoted $\mathbf{p}(\cdot | y) \ll \mathbf{p}$), and its Radon–Nikodym derivative is

$$\frac{d\mathbf{p}(\cdot | y)}{d\mathbf{p}}(f) = \frac{L(y | f)}{Z(y)} \quad \mathbf{p}\text{-a.s.}$$

We call $\mathbf{p}(\cdot | y)$ the *posterior measure* on the function space.

We note here that there is no appeal to “posterior densities” on $\mathcal{F}$. The posterior is always defined *relative to the prior* via the measurable likelihood functional.

17.2.3 Finite-Dimensional (Evaluation) Form

If the likelihood depends only on the training evaluations, i.e.

$$L(y | f) = \tilde{L}(y | \text{eval}_{\mathbf{x}_{\text{tr}}}(f)),$$

define the finite-dimensional evidence

$$\tilde{Z}(y) := \int_{\mathbb{R}^n} \tilde{L}(y | \mathbf{u}) d\mathbf{p}_{\mathbf{x}_{\text{tr}}}(\mathbf{u}).$$

By the definition of a pushforward integral, $\tilde{Z}(y) = Z(y)$. Thus, in addition to

$$\frac{d\mathbf{p}(\cdot | y)}{d\mathbf{p}}(f) = \frac{\tilde{L}(y | \text{eval}_{\mathbf{x}_{\text{tr}}}(f))}{\tilde{Z}(y)}, \quad \mathbf{p}\text{-a.s.},$$

we also have

$$\frac{d\mathbf{p}_{\mathbf{x}_{\text{tr}}}(\cdot | y)}{d\mathbf{p}_{\mathbf{x}_{\text{tr}}}}(\mathbf{u}) = \frac{\tilde{L}(y | \mathbf{u})}{\tilde{Z}(y)}, \quad \mathbf{p}_{\mathbf{x}_{\text{tr}}}\text{-a.s.} \quad (17.1)$$

Thus the posterior marginal at $\mathbf{x}_{\text{tr}}$ is obtained using a familiar finite-dimensional calculation: by reweighting the corresponding prior marginal $\mathbf{p}_{\mathbf{x}_{\text{tr}}}$ with the likelihood and renormalising. However, although (17.1) gives the correct posterior marginal at the training inputs, that single marginal does not determine the full posterior measure $\mathbf{p}(\cdot | y)$ on $\mathcal{F}$.

17.3 Function-Space KL and Variational Objectives

17.3.1 Measure-theoretic KL divergence

The Kullback–Leibler divergence between two probability measures $\mathbf{q}$ and $\mathbf{p}$ on $(\mathcal{F}, \mathfrak{F})$ is

$$\text{KL}(\mathbf{q} \| \mathbf{p}) = \begin{cases} \int_{\mathcal{F}} \log\left(\frac{d\mathbf{q}}{d\mathbf{p}}\right) d\mathbf{q}, & \text{if } \mathbf{q} \ll \mathbf{p}, \\ +\infty, & \text{otherwise,} \end{cases}$$

where $\frac{d\mathbf{q}}{d\mathbf{p}}$ denotes the Radon–Nikodym derivative of $\mathbf{q}$ with respect to $\mathbf{p}$. We note that $\text{KL}(\mathbf{q} \| \mathbf{p})$ may be infinite even when $\mathbf{q} \ll \mathbf{p}$. Absolute continuity guarantees the existence of the Radon–Nikodym derivative $\frac{d\mathbf{q}}{d\mathbf{p}}$, but imposes no integrability conditions on $\log\left(\frac{d\mathbf{q}}{d\mathbf{p}}\right)$ under $\mathbf{q}$.

17.3.2 Function-space ELBO

For fixed data y , we henceforth denote the posterior measure $\mathbf{p}(\cdot | y)$ by $\hat{\mathbf{p}}$. Suppose $\mathbf{q} \ll \mathbf{p}$, $\mathbf{q} \ll \hat{\mathbf{p}}$ and both $\text{KL}(\mathbf{q} \| \hat{\mathbf{p}})$ and $\text{KL}(\mathbf{q} \| \mathbf{p})$ are finite. Then, we can apply the chain rule for Radon–Nikodym derivatives:

$$\begin{aligned} \text{KL}(\mathbf{q} \| \hat{\mathbf{p}}) &= \int \log\left(\frac{d\mathbf{q}}{d\hat{\mathbf{p}}}\right) d\mathbf{q} = \int \log\left(\frac{d\mathbf{q}}{d\mathbf{p}}\right) d\mathbf{q} - \int \log\left(\frac{d\hat{\mathbf{p}}}{d\mathbf{p}}\right) d\mathbf{q} \\ &= \int \log\left(\frac{d\mathbf{q}}{d\mathbf{p}}\right) d\mathbf{q} - \int \log\left(\frac{L(y | f)}{Z(y)}\right) d\mathbf{q} \\ &= \underbrace{\text{KL}(\mathbf{q} \| \mathbf{p})}_{\text{regulariser (KL to prior)}} - \underbrace{\mathbb{E}_{\mathbf{q}}[\log L(y | f)]}_{\text{data fit}} + \log Z(y). \end{aligned}$$

Rearranging yields the evidence lower bound

$$\log Z(y) \geq \mathbb{E}_{\mathbf{q}}[\log L(y | f)] - \text{KL}(\mathbf{q} \| \mathbf{p}) =: \text{ELBO}(\mathbf{q}) \quad (17.2)$$

with equality if and only if $\mathbf{q} = \hat{\mathbf{p}}$. This is the *variational objective in function space*. We wish to consider problems of the form

$$\max_{\mathbf{q} \in \mathcal{Q}} \text{ELBO}(\mathbf{q}) \quad (17.3)$$

for some variational family $\mathcal{Q}$.

17.3.3 Well-posedness of ELBO maximisation

A simple way to ensure that the ELBO is well defined throughout the variational family $\mathcal{Q}$ is to require that every $\mathbf{q} \in \mathcal{Q}$ satisfies $\mathbf{q} \ll \mathbf{p}$ and that both $\text{KL}(\mathbf{q} \| \mathbf{p})$ and $\mathbb{E}_{\mathbf{q}}[\log L(y | f)]$ are finite. This is not always straightforward to ensure. For example, it is possible that the KL to the prior is finite, i.e. $\mathbf{q} \ll \mathbf{p}$ and $\text{KL}(\mathbf{q} \| \mathbf{p}) < \infty$, while the KL to the posterior is not. Indeed, if $\mathbf{q}$ puts positive mass where $d\hat{\mathbf{p}}/d\mathbf{p} = 0$, then $\text{KL}(\mathbf{q} \| \hat{\mathbf{p}}) = \infty$ and

$\log L(y | f) = -\infty$ on a set of positive $\mathbf{q}$ -measure. If, in addition, the positive part of $\log L(y | f)$ is integrable, then $\mathbb{E}_{\mathbf{q}}[\log L(y | f)] = -\infty$ and the lower bound (17.2) is vacuous; otherwise the expected log-likelihood may even be undefined. Candidates with an undefined expected log-likelihood must be excluded, whereas a family for which every candidate has ELBO equal to $-\infty$ also yields an ill-posed maximisation problem.

If, in addition to $0 < Z(y) < \infty$, we assume

$$0 < L(y | f) < \infty \quad \text{for } \mathbf{p}\text{-almost every } f,$$

then

$$\frac{d\hat{\mathbf{p}}}{d\mathbf{p}}(f) = \frac{L(y | f)}{Z(y)} > 0 \quad \mathbf{p}\text{-a.s.},$$

so $\hat{\mathbf{p}}$ and $\mathbf{p}$ are equivalent measures. Consequently, any $\mathbf{q} \ll \mathbf{p}$ also satisfies $\mathbf{q} \ll \hat{\mathbf{p}}$, although this equivalence alone does not make the KL divergences, the expected log-likelihood, or the ELBO finite; hence particular care needs to be taken in choosing the variational family. A sufficient condition is that for every variational law under consideration,

$$\text{KL}(\mathbf{q} \parallel \mathbf{p}) < \infty, \quad \mathbb{E}_{\mathbf{q}}[|\log L(y | f)|] < \infty.$$

Leaving aside the difficulty of whether the ELBO objective is well defined, we still have a challenge of efficiently estimating it and computing its gradients for a given variational family. The next section discusses a supremum characterisation of the function-space KL divergence that motivates practical approaches to ELBO maximisation.

17.4 Supremum characterisation of the ELBO

Let $\mathbf{p}$ and $\mathbf{q}$ be arbitrary probability measures on $\mathbb{R}^{\mathcal{X}}$ equipped with the cylinder σ -algebra generated by all finite evaluation maps. For a finite tuple $\mathbf{z} = (z_1, \dots, z_m)$, denote their pushforward marginals by $\mathbf{p}_{\mathbf{z}}$ and $\mathbf{q}_{\mathbf{z}}$. On this measurable space, with no additional topology or sample-path regularity assumptions, [1576] show that

$$\text{KL}(\mathbf{q} \parallel \mathbf{p}) = \sup_{m \in \mathbb{N}, \mathbf{z} \in \mathcal{X}^m} \text{KL}(\mathbf{q}_{\mathbf{z}} \parallel \mathbf{p}_{\mathbf{z}}). \quad (17.4)$$

The identity follows because the cylinder σ -algebra contains precisely the information generated by finite evaluations. Note that one should not transfer the identity without further argument to a larger Borel σ -algebra or to other function spaces.

For the supervised likelihood introduced above, define

$$\text{ELBO}_{\mathbf{z}}(\mathbf{q}) := \mathbb{E}_{\mathbf{q}} \left[\log \tilde{L}(y | \text{eval}_{\mathbf{x}_{\text{tr}}}(f)) \right] - \text{KL}(\mathbf{q}_{\mathbf{z}} \parallel \mathbf{p}_{\mathbf{z}}).$$

Substitution in the function-space ELBO gives

$$\text{ELBO}(\mathbf{q}) = \inf_{m \in \mathbb{N}, \mathbf{z} \in \mathcal{X}^m} \text{ELBO}_{\mathbf{z}}(\mathbf{q}).$$

Note that the data-fit term depends on the fixed training tuple $\mathbf{x}_{\text{tr}}$, not on the auxiliary tuple $\mathbf{z}$ over which the infimum is taken. The infimum need not be attained, however, and its value may still be $-\infty$ when the function-space KL is infinite.

17.4.1 Adversarial functional VI

Restricting the tuple $\mathbf{z}$ to a fixed size $m \in \mathbb{N}$ gives the finite-dimensional saddle problem

$$\max_{\mathbf{q}} \min_{\mathbf{z} \in \mathcal{X}^m} \text{ELBO}_{\mathbf{z}}(\mathbf{q}).$$

Optimisation alternates between minimising over $\mathbf{z}$ and maximising over $\mathbf{q}$, in analogy with a capacity-limited adversary in a generative adversarial network [650]. This finite-evaluation objective is not generally an ELBO: it retains an evidence-lower-bound interpretation only when $\mathbf{z}$ contains every input whose function value enters the likelihood, here all of $\mathbf{x}_{\text{tr}}$ [1576]. As [1576] point out, the adversarial scheme can generalise poorly. Its selected evaluation locations are likely to remain close to the training data, and the prior enters the objective only through the KL term, so prior information may not be used effectively for extrapolation.

17.4.2 Sampling-based functional VI

To ensure that the KL term constrains behaviour in prediction regions, [1576] augment a random subsample of training inputs with evaluation locations drawn from the prediction domain. Let Π_{tr} be a specified distribution on finite tuples sampled from $\{x_1, \dots, x_n\}$, and let ν^* be a user-chosen probability distribution on a prediction domain. Write $\mathbf{x}_{\text{sub}} \sim \Pi_{\text{tr}}$ for a random training-input tuple, $\mathbf{z}^* \sim (\nu^*)^{\otimes m_*}$ for m_* independent prediction-domain locations, and $\oplus$ for tuple concatenation. The sampling-based objective of [1576] is

$$\max_{\mathbf{q}} \mathbb{E}_{\mathbf{x}_{\text{sub}} \sim \Pi_{\text{tr}}} \mathbb{E}_{\mathbf{z}^* \sim (\nu^*)^{\otimes m_*}} [\text{ELBO}_{\mathbf{x}_{\text{sub}} \oplus \mathbf{z}^*}(\mathbf{q})].$$

The data-fit term is always evaluated at the full training tuple $\mathbf{x}_{\text{tr}}$; only the KL term uses the sampled tuple. The sampling scheme thereby encourages agreement with the prior both on and off the training set. However, we have replaced the infimum with expectation, so the connection to the function-space ELBO is no longer direct. In the special case where the true posterior is a GP and every evaluation tuple contains all the training inputs, the true posterior maximises both the expected finite-measurement objective and the full function-space objective, providing a justification for using expectation; see [1576] for details. The new criterion makes it possible to combine structured GP priors (e.g., periodic plus RBF kernels) with a BNN variational family. In experiments reported by [1576], this combination recovered periodic structure and extrapolated comparably to exact GPs in examples where standard weight-space VI failed to extrapolate sensibly. The study also reports competitive and scalable results on UCI data sets and applies the method to contextual bandits. In those studied settings, the sampling-based scheme yielded better overall performance than the adversarial scheme.

17.4.3 Grid-functional divergence

Sampling-based functional VI replaces the supremum in (17.4) by an expectation of marginal KL divergences but keeps the size of the sampled tuples fixed. [1096] take this idea further and also randomise the tuple size. Let M have a probability mass function π on $\mathbb{N}$ with $\pi(m) > 0$ for every m , and, conditionally on $M = m$, draw $\mathbf{z}_M \sim \nu_m^*$ on $\mathcal{X}^m$. Define the augmented grid $\mathbf{u}_M = \mathbf{x}_{\text{tr}} \oplus \mathbf{z}_M$ and

$$\text{KL}_{\text{grid}}(\mathbf{q} \parallel \mathbf{p}) := \mathbb{E}_{M \sim \pi} \mathbb{E}_{\mathbf{z}_M \sim \nu_M^*} [\text{KL}(\mathbf{q}_{\mathbf{u}_M} \parallel \mathbf{p}_{\mathbf{u}_M})]. \quad (17.5)$$

In the setting of [1096], where $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$, suppose each ν_m^* has full support on $\mathcal{X}^m$. Since $\pi(m) > 0$ for every m , Proposition 1 then gives $\text{KL}_{\text{grid}}(\mathbf{q} \parallel \mathbf{p}) = 0$ if and only if $\mathbf{q} = \mathbf{p}$. If $M = m$ is fixed, only the marginals on augmented tuples $\mathbf{x}_{\text{tr}} \oplus \mathbf{z}_m$ are compared; distinct process laws can agree on all such marginals while differing on larger tuples. Depending on the choice of π , the expectation can remain finite even when the supremum KL is infinite. Moreover, because every $\mathbf{u}_M$ contains all training inputs, Proposition 3 of [1096] ensures that we still obtain a lower bound on the evidence,

$$\log Z(y) \geq \mathbb{E}_{\mathbf{q}} \left[\log \tilde{L}(y \mid \text{eval}_{\mathbf{x}_{\text{tr}}}(f)) \right] - \text{KL}_{\text{grid}}(\mathbf{q} \parallel \mathbf{p}). \quad (17.6)$$

The paper combines this objective with stochastic process generators, a flexible non-Gaussian variational family, and analytic and mini-batch estimators. In the experiments reported by [1096], the resulting method reproduced the qualitative posterior structure of piecewise-constant and piecewise-linear implicit priors and performed strongly against the selected baselines on UCI regression, contextual bandits, and image classification with out-of-distribution detection.

17.5 Parameter pushforward measures

Let $\Theta \subset \mathbb{R}^d$ be a finite-dimensional parameter space with Borel σ -algebra $\mathcal{B}(\Theta)$. Let $g : \Theta \rightarrow \mathcal{F}$ be a measurable parameter-to-function map, $g(\theta) = f(\cdot; \theta)$. Given a *parameter prior* P_{Θ} and a *variational parameter law* Q_{Θ} on $(\Theta, \mathcal{B}(\Theta))$, define the induced pushforward measures

$$\mathbf{p}^g := g_{\#} P_{\Theta}, \quad \mathbf{q}^g := g_{\#} Q_{\Theta},$$

so that $\mathbf{p}^g(A) = P_{\Theta}(\{\theta : g(\theta) \in A\})$ and likewise for $\mathbf{q}^g$.

17.5.1 Data-processing inequality

For any measurable g , the data-processing inequality (cf. e.g. [1339]) gives

$$\boxed{\text{KL}(\mathbf{q}^g \parallel \mathbf{p}^g) \leq \text{KL}(Q_\Theta \parallel P_\Theta)}. \quad (17.7)$$

Suppose $Q_\Theta \ll P_\Theta$ and the right-hand side is finite, and let $h = dQ_\Theta/dP_\Theta$. Equality holds precisely when h is measurable with respect to the σ -algebra generated by g , equivalently when $h = \varphi \circ g$ P_Θ -a.s. for some measurable φ . Thus, a sufficient condition for equality is P_Θ -almost-sure injectivity of g . In general, however, g is many-to-one and equality typically does not hold.

Thus, if one chooses the *function-space prior* to be the *same* BNN-induced law $\mathbf{p}^g$ and approximates it with $\mathbf{q}^g$ induced by a different parameter law, the function-space KL cannot exceed the parameter-space KL. For the same expected log-likelihood, the function-space ELBO is therefore at least as large. This formalises why working directly in function space can give *tighter* objectives than parameter-space VI when g is many-to-one (and, of course, deep neural network models yield many-to-one mappings). This comparison of objective values does not, by itself, guarantee a better predictive approximation, but a tighter function-space objective avoids penalising discrepancies between parameter laws that induce the same function law, thereby aligning the regularisation more closely with the predictive object of interest.

17.5.2 Estimating function-space KL for parameter pushforwards

The function-space KL in (17.7) is finite whenever $\text{KL}(Q_\Theta \parallel P_\Theta) < \infty$. It is nevertheless a divergence between generally non-standard pushforward laws and still requires approximation.

One tractable approximation is proposed by [1435]. For a finite tuple $\mathbf{z} = (z_1, \dots, z_m) \in \mathcal{X}^m$, write $g_{\mathbf{z}}(\theta) = \text{eval}_{\mathbf{z}}(g(\theta)) \in \mathbb{R}^m$. For $G \in \{P_\Theta, Q_\Theta\}$, define the parameter mean and covariance

$$m_G := \mathbb{E}_{\theta \sim G}[\theta], \quad S_G := \text{Cov}_{\theta \sim G}(\theta).$$

Assume that $g_{\mathbf{z}}$ is differentiable at m_G and define its Jacobian by

$$J_{\mathbf{z}}(m_G) := \left[\frac{\partial g(z_i, \theta)}{\partial \theta_j} \Big|_{\theta=m_G} \right]_{\substack{1 \leq i \leq m \\ 1 \leq j \leq d}} \in \mathbb{R}^{m \times d}.$$

Linearise $g_{\mathbf{z}}$ at m_G :

$$g_{\mathbf{z}}(\theta) \approx g_{\mathbf{z}}(m_G) + J_{\mathbf{z}}(m_G)(\theta - m_G).$$

This is the same local parameter-to-function construction used in linearised Laplace approximations; see Chapter 8. It gives the Gaussian marginal approximations

$$(\mathbf{p}^g)_{\mathbf{z}} \approx \mathcal{N}(\mu_P(\mathbf{z}), \Sigma_P(\mathbf{z})), \quad (\mathbf{q}^g)_{\mathbf{z}} \approx \mathcal{N}(\mu_Q(\mathbf{z}), \Sigma_Q(\mathbf{z})),$$

where

$$\mu_G(\mathbf{z}) := g_{\mathbf{z}}(m_G), \quad \Sigma_G(\mathbf{z}) := J_{\mathbf{z}}(m_G) S_G J_{\mathbf{z}}(m_G)^\top.$$

Thus m_G, S_G are moments in parameter space, whereas μ_G, Σ_G are moments of the linearised function values.

Replacing the supremum in (17.4) by a maximum over user-chosen regularisation tuples $\{\mathbf{z}^{(s)}\}_{s=1}^S$, the surrogate function-space objective becomes

$$\begin{aligned} \tilde{\mathcal{F}}(Q_\Theta) &:= \mathbb{E}_{\theta \sim Q_\Theta} \left[\log \tilde{L}(y \mid g_{\mathbf{x}_{\text{tr}}}(\theta)) \right] \\ &\quad - \max_{1 \leq s \leq S} \text{KL} \left(\mathcal{N}(\mu_Q(\mathbf{z}^{(s)}), \Sigma_Q(\mathbf{z}^{(s)})) \parallel \mathcal{N}(\mu_P(\mathbf{z}^{(s)}), \Sigma_P(\mathbf{z}^{(s)})) \right). \end{aligned}$$

The prior moments m_P, S_P are fixed by P_Θ , while the moments of Q_Θ are variational quantities. In practice, samples from Q_Θ are reparameterised and the Jacobians are obtained by automatic differentiation. By construction, (i) the data-processing inequality ensures a well-posed function-space regulariser when the parameter-space KL is finite, and (ii) the linearisation supplies closed-form KL terms at each $\mathbf{z}$, yielding a scalable estimator that preserves the function-space viewpoint while remaining compatible with BNN pushforwards.

17.5.3 Interpretable priors and the singularity issue

Often we do not want the parameter pushforward $\mathbf{p}^g$ as the prior – it is not clear what prior beliefs on neural network weights represent. Instead, one may specify a non-degenerate GP prior $\mathbf{p}$ whose covariance kernel expresses structural assumptions about the unknown function, while retaining the BNN variational law $\mathbf{q}^g$. Function-space priors are discussed further in Chapter 31.

This choice falls outside the common-map setting of (17.7). It also creates a dimension mismatch. Fix $\mathbf{z} = (z_1, \dots, z_m)$ and consider

$$g_{\mathbf{z}} : \theta \mapsto (g(z_1, \theta), \dots, g(z_m, \theta)) \in \mathbb{R}^m.$$

Assume that $g_{\mathbf{z}}$ is locally Lipschitz, as is the case for standard finite neural networks with locally Lipschitz activations. If $m > d$, its image is a countable union of sets with Hausdorff dimension at most d , and therefore has m -dimensional Lebesgue measure zero. The marginal $\mathbf{q}_{\mathbf{z}}^g = (g_{\mathbf{z}})_{\#} Q_{\Theta}$ assigns probability one to this image. By contrast, if the GP covariance matrix at $\mathbf{z}$ is positive definite, $\mathbf{p}_{\mathbf{z}}$ has a Lebesgue density on $\mathbb{R}^m$ and assigns probability zero to the image. For geometric intuition, a one-parameter network evaluated at two inputs traces a curve in the plane; a non-degenerate bivariate Gaussian assigns zero probability to that curve.

Hence $\mathbf{q}_{\mathbf{z}}^g$ and $\mathbf{p}_{\mathbf{z}}$ are mutually singular for such a tuple, and the cylinder-space identity (17.4) yields

$$\text{KL}(\mathbf{q}^g \parallel \mathbf{p}) = \sup_{m, \mathbf{z}} \text{KL}(\mathbf{q}_{\mathbf{z}}^g \parallel \mathbf{p}_{\mathbf{z}}) = +\infty.$$

KL-based objectives are hence incompatible with the setting where we wish to posit an interpretable GP prior but to use a flexible neural network parameterisation of the posterior. This motivates considering discrepancies that do not require absolute continuity.

17.6 Generalised Variational Objectives

17.6.1 Motivation

In function-space VI with interpretable priors, the KL-based ELBO may equal $-\infty$ because the variational law and the prior may be mutually singular; in particular, the KL divergences between a parametric pushforward and a GP measure can be infinite in both directions. Even if we instead used a variational measure that is absolutely continuous with respect to the prior, there is generally no easy way to estimate KL in infinite-dimensional spaces. This motivates variational objectives with better analytical structure. Consequently, in this section we seek alternative discrepancies that remain finite and tractable for rich function-space distributions.

17.6.2 Generalised variational objective (GVO)

Let $(\mathcal{F}, \mathfrak{F})$ be a measurable function space, let $\mathbf{p}$ be a prior, and let $L(y \mid f)$ be a positive likelihood. Following [904], replace the KL regularisation by a discrepancy D and define a generalised variational objective (GVO) as:

$$\mathcal{J}_D(\mathbf{q}) := \mathbb{E}_{\mathbf{q}}[\log L(y \mid f)] - D(\mathbf{q}, \mathbf{p}).$$

When $D = \text{KL}$, this is the ordinary ELBO. Taking $D = \text{KL}_{\text{grid}}$ as defined in (17.5) yields the approach of [1096]. The linearised Gaussian approximation of [1435] in Section 17.5.2 can likewise be viewed as a generalised objective over parameter laws, with the maximum of its finite-marginal Gaussian KL terms serving as a surrogate discrepancy. For a general D , $\mathcal{J}_D$ need not lower-bound the evidence, and its optimiser need not be the ordinary Bayesian posterior. It instead defines a generalised update whose interpretation depends on D .

17.6.3 Wasserstein-2 GVO for Gaussian measures on function spaces

A particularly attractive choice for D in function space is the squared 2-Wasserstein distance W_2^2 , which remains finite for any two Gaussian measures in a separable Hilbert space and has a closed form expression. Moreover, the 2-Wasserstein distance is a genuine metric and measures the geometry of transporting one probability law to another.

Background on Gaussian measures in Hilbert spaces can be found in [352, 943]; Gaussian measures on more general linear spaces such as Banach or Fréchet spaces are treated by [180]. Gaussian measures are closely related to Gaussian processes, but differ in that they are explicitly defined on a specific function space, whereas trajectories of a Gaussian process can be embedded into multiple function spaces.

Assume that $\mathcal{F}$ is a separable Hilbert space, such as $L^2(\mathcal{X}, \rho)$ for a finite measure ρ or a reproducing kernel Hilbert space. An operator A on $\mathcal{F}$ is *Hilbert–Schmidt* if $\sum_i \|Ae_i\|_{\mathcal{F}}^2 < \infty$ for an orthonormal basis $(e_i)_i$. A positive operator C is *trace-class* if $\text{tr}(C) = \sum_i \langle Ce_i, e_i \rangle_{\mathcal{F}} < \infty$. Every positive self-adjoint operator C has a unique positive square root, and C is trace-class if and only if $C^{1/2}$ is Hilbert–Schmidt. Covariance operators of Gaussian measures on $\mathcal{F}$ are always positive, self-adjoint, and trace-class.

Suppose $\mathbf{p} = \mathcal{N}(m_{\mathbf{p}}, C_{\mathbf{p}})$ and $\mathbf{q} = \mathcal{N}(m_{\mathbf{q}}, C_{\mathbf{q}})$ are Gaussian measures on $\mathcal{F}$, with $m_{\mathbf{p}}, m_{\mathbf{q}} \in \mathcal{F}$. Then

$$W_2^2(\mathbf{q}, \mathbf{p}) = \|m_{\mathbf{q}} - m_{\mathbf{p}}\|_{\mathcal{F}}^2 + \text{tr}\left(C_{\mathbf{q}} + C_{\mathbf{p}} - 2(C_{\mathbf{p}}^{1/2}C_{\mathbf{q}}C_{\mathbf{p}}^{1/2})^{1/2}\right). \quad (17.8)$$

Let $B = C_{\mathbf{p}}^{1/2}C_{\mathbf{q}}C_{\mathbf{p}}^{1/2}$. Since $C_{\mathbf{p}}^{1/2}$ and $C_{\mathbf{q}}^{1/2}$ are Hilbert–Schmidt,

$$\text{tr}(B^{1/2}) = \|C_{\mathbf{q}}^{1/2}C_{\mathbf{p}}^{1/2}\|_{S_1} \leq \|C_{\mathbf{q}}^{1/2}\|_{S_2}\|C_{\mathbf{p}}^{1/2}\|_{S_2} < \infty,$$

where S_1 and S_2 denote the trace-class and Hilbert–Schmidt norms. Thus, $W_2(\mathbf{q}, \mathbf{p})$ is finite without any absolute-continuity assumptions.

The Feldman–Hájek theorem states that two Gaussian measures on a separable Hilbert space are either equivalent or mutually singular [352, 943]. Equivalence holds only under substantially stricter compatibility conditions. In particular, on the common covariance support, the relative covariance perturbation $S = C_{\mathbf{p}}^{-1/2}C_{\mathbf{q}}C_{\mathbf{p}}^{-1/2} - I$ must admit a Hilbert–Schmidt extension, in addition to conditions on the covariance ranges and the difference of the means. If these conditions fail, the measures are mutually singular and their KL divergence is infinite. This is in contrast to the automatic trace-class property needed for the Wasserstein cross term in (17.8).

Note that we have required both the prior and variational function laws to be Gaussian measures. Non-Gaussian likelihoods can still enter through $\mathbb{E}_{\mathbf{q}}[\log L]$, but the variational law cannot represent non-Gaussian posteriors. Hence flexibility needs to be introduced through the parametrisation of the mean function and the covariance operator.

17.6.4 Generalised Wasserstein inference with a neural mean (GWI-net)

Having selected the Wasserstein-2 distance, we need to parameterise the variational law $\mathbf{q}$. One possibility is to approximate a parameter pushforward $\mathbf{q}^g = g_{\#}Q_{\Theta}$ by a Gaussian measure on $\mathcal{F}$. Alternatively, the generalised Wasserstein inference network (GWI-net) of [1755] uses a hybrid Gaussian law whose mean $m_{\mathbf{q}} = f(\cdot; \theta)$ is a deep neural network and whose covariance has a sparse variational GP form. Here θ is a deterministic variational parameter of the mean function, not a random BNN weight.

We focus on $\mathcal{F} = L^2(\mathcal{X}, \rho)$, where ρ is a chosen reference probability measure (typically the input law).

Prior (Gaussian measure). Let $\mathbf{p} = \mathcal{N}(m_{\mathbf{p}}, C_{\mathbf{p}})$, with

$$(C_{\mathbf{p}}\phi)(x) = \int_{\mathcal{X}} k(x, x')\phi(x') d\rho(x'), \quad \phi \in \mathcal{F},$$

where $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a symmetric positive-definite kernel such that $C_{\mathbf{p}}$ is trace-class.

Variational law (Gaussian measure). Let $\mathbf{q} = \mathcal{N}(m_{\mathbf{q}}, C_{\mathbf{q}})$, with

$$(C_{\mathbf{q}}\phi)(x) = \int_{\mathcal{X}} r(x, x')\phi(x') d\rho(x'), \quad \phi \in \mathcal{F},$$

where $r : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a symmetric positive-definite kernel parameterised by additional variational parameters.

The neural mean, covariance parameters, inducing locations, and any likelihood parameters are trained by maximising the expected log-likelihood minus an empirical approximation of the squared Wasserstein distance in (17.8). For Gaussian regression, $y_i | F \sim \mathcal{N}(F(x_i), \sigma^2)$, the expected log-likelihood is analytic; for non-Gaussian likelihoods it can be approximated by quadrature or Monte Carlo. The fitted Gaussian law supplies predictive means and covariances, with the observation likelihood providing any additional predictive noise.

To estimate the Wasserstein distance, note that

$$\begin{aligned} \|m_{\mathbf{p}} - m_{\mathbf{q}}\|_{\mathcal{F}}^2 &= \int_{\mathcal{X}} (m_{\mathbf{p}}(x) - m_{\mathbf{q}}(x))^2 d\rho(x), \\ \text{tr}(C_{\mathbf{p}}) &= \int_{\mathcal{X}} k(x, x) d\rho(x), \quad \text{tr}(C_{\mathbf{q}}) = \int_{\mathcal{X}} r(x, x) d\rho(x), \end{aligned}$$

under the usual trace-kernel conditions [208]. Replacing ρ by the empirical design measure $\hat{\rho}_n = n^{-1} \sum_{i=1}^n \delta_{x_i}$ turns these terms into sample averages. These averages are Monte Carlo-unbiased for fixed integrands when $x_1, \dots, x_n$ are i.i.d. from ρ ; for a fixed design, they instead define a discretisation at the training inputs.

The remaining cross term is estimated from a spectral subsample $\mathbf{x}_S = (x_{s_j})_{j=1}^{n_S}$ of $n_S \ll n$ training inputs. Write

$$r(\mathbf{x}_S, \mathbf{x}_{\text{tr}}) = (r(x_{s_j}, x_i))_{j,i}, \quad k(\mathbf{x}_{\text{tr}}, \mathbf{x}_S) = (k(x_i, x_{s_j}))_{i,j}.$$

The approximation of [1755] is

$$\begin{aligned} \widehat{W}_2^2 = & \frac{1}{n} \sum_{i=1}^n (m_{\mathbf{p}}(x_i) - m_{\mathbf{q}}(x_i))^2 + \frac{1}{n} \sum_{i=1}^n k(x_i, x_i) \\ & + \frac{1}{n} \sum_{i=1}^n r(x_i, x_i) - \frac{2}{\sqrt{nn_S}} \sum_{j=1}^{n_S} \sqrt{\lambda_j(r(\mathbf{x}_S, \mathbf{x}_{\text{tr}})k(\mathbf{x}_{\text{tr}}, \mathbf{x}_S))}. \end{aligned} \quad (17.9)$$

The matrix spectrum is an empirical Nyström approximation to the covariance-operator composition. Increasing n_S improves spectral resolution at greater matrix cost; [1755] gives an empirical study under rapid spectral decay but no unbiasedness or consistency results have been shown. With the empirical design measure, behaviour away from the training inputs is determined indirectly by the kernels and neural mean; additional points drawn from a broader reference measure can instead be used in the discretisation.

Many parameterisations of r are possible. Following [1755], let $\mathbf{Z} = (z_1, \dots, z_M) \in \mathcal{X}^M$ be inducing locations, and define

$$\mathbf{k}_Z(x) := [k(z_1, x), \dots, k(z_M, x)]^\top, \quad \mathbf{K}_{ZZ} := (k(z_i, z_j))_{i,j=1}^M.$$

For a free lower-triangular factor $L \in \mathbb{R}^{M \times M}$ and nonsingular $\mathbf{K}_{ZZ}$, set

$$r(x, x') = k(x, x') - \mathbf{k}_Z(x)^\top \mathbf{K}_{ZZ}^{-1} \mathbf{k}_Z(x') + \mathbf{k}_Z(x)^\top L L^\top \mathbf{k}_Z(x'). \quad (17.10)$$

This is the sparse variational GP covariance construction of [1615]; stochastic mini-batch variants are developed by [737, 738], and Chapter 38 gives further details on inducing-point methods.

In the experiments of [1755], GWI-net was evaluated on synthetic examples, UCI regression, image classification, and out-of-distribution detection. The paper reports the best mean rank in its UCI comparison, strong classification results, and mixed out-of-distribution performance across datasets. The method remains limited by its Gaussian variational family, the spectral-decay approximation, and numerical sensitivity in the inducing-point parameterisation.

17.7 Discussion

Function-space variational inference reframes Bayesian deep learning at the level where inductive bias lives: distributions over functions. This perspective yields a measure-theoretic ELBO, clarifies why weight-space KLs can mislead in infinite dimensions, and motivates workable surrogates—from supremum-over-marginals and sampling-based objectives to grid-functional divergences. It also explains, via data-processing, when parameter pushforwards are overly restrictive and how linearised, finite-marginal Gaussian approximations can recover tractable regularisers. Most broadly, generalised objectives can replace KL with alternative discrepancies, such as grid-based KL or Wasserstein-2, that remain finite and computable in a broader range of settings, enabling hybrid variational laws that pair neural posteriors with interpretable GP-like priors. These approaches pave the way towards a toolkit that preserves Bayesian discipline while delivering the extrapolation, calibration, and scalability modern applications demand. In practice, however, finite discretisations, Gaussian variational restrictions, empirical approximations and numerical sensitivity limit these methods, and the current literature does not establish general guarantees of extrapolation, calibration, or scalability. Looking ahead, the key opportunities are tighter estimators of function-space discrepancies, richer variational family parameterisations, and principled integration of domain priors about the objects of interest—such as invariances, monotonicity or convexity, and conservation constraints—encoded directly in function space to steer inference toward scientifically plausible solutions.

Chapter 18

Sequential variational inference

18.1 Introduction

This chapter is concerned with sequential or online Bayesian inference of the parameters of a deep neural network. That is, we want to recursively compute $p(\theta_t | \mathcal{D}_{1:t})$ as the data streams in, where $\mathcal{D}_t = (x_t, y_t)$ is the t^{th} datapoint, and θ_t are the parameters at time t . This is an online version of the standard Bayesian deep learning (BDL) problem, and is useful for settings where we need to take a decision at each step (e.g., contextual bandits [466]), and/or for settings where the data generating process is non-stationary (e.g., continual learning [1020, 1860]). We approach this problem from the perspective of Bayesian filtering (as opposed to smoothing, which works offline to compute $p(\theta_t | \mathcal{D}_{1:T})$ where $\mathcal{D}_{1:T}$ is the full dataset). If the parameters are assumed to be constant over time — which we shall initially assume, for notational simplicity — we can replace θ_t with θ .

The quantity $p(\theta | \mathcal{D}_{1:t})$ is known as the belief state, since it represents our current beliefs about the unknown parameters given the data so far. (It is also known as the filtering distribution, as mentioned in Section 1.6.) This can be computed using repeated applications of Bayes' rule:

$$p(\theta | \mathcal{D}_{1:t}) = \frac{p(y_t | x_t, \theta)p(\theta | \mathcal{D}_{1:t-1})}{p(y_t | x_t, \mathcal{D}_{1:t-1})} \quad (18.1)$$

where the prior $p(\theta | \mathcal{D}_{1:t-1})$ is the previous belief state, and the denominator is given by

$$p(y_t | x_t, \mathcal{D}_{1:t-1}) \stackrel{\text{def}}{=} Z_t = \int p(y_t | x_t, \theta)p(\theta | \mathcal{D}_{1:t-1}) d\theta \quad (18.2)$$

Note that, in the batch (offline) setting, the product of the normalising factors across steps corresponds to the marginal likelihood of the data:

$$Z(\mathcal{D}) = p(y_{1:N} | x_{1:N}) = \prod_{n=1}^N Z_n \quad (18.3)$$

The key challenge we focus on in this chapter is how to efficiently compute the new belief state $p(\theta | \mathcal{D}_{1:t})$ at each step, when the unknown parameters $\theta \in \mathbb{R}^P$ are high dimensional, and where the likelihood is not conjugate to the prior. To make the problem computationally efficient (which is particularly important in online settings), we will focus on variational inference (VI) methods, which were introduced (in a batch setting) in Chapter 12. We will represent our approximate belief state at each step by $q_t = q(\theta | \mathcal{D}_{1:t})$. We will focus on the special case where this variational distribution is a Gaussian, $q_t = \mathcal{N}(\cdot | \mu_t, \Sigma_t)$, where μ_t is the posterior mean and Σ_t is the posterior covariance. (In many cases, it is more convenient to work with the posterior precision matrix, denoted $\Lambda_t = \Sigma_t^{-1}$.) We will discuss a variety of methods for efficiently updating these variational parameters, which will often require imposing additional assumptions on Σ_t (or Λ_t), in order to trade off expressivity with efficiency. We will also make connections between these Bayesian inference methods and optimisation algorithms, such as natural gradient descent (discussed in Chapter 15) and other second-order methods.

18.2 State Space Models

The methods we discuss in this chapter can be applied to many kinds of state estimation problems, not just estimating neural network parameters. We will therefore start with a general presentation before specialising it

to the BDL case. This will make it easier to see connections to the broader literature on Bayesian filtering (e.g., Särkkä and Svensson [1461]).

18.2.1 Modelling assumptions

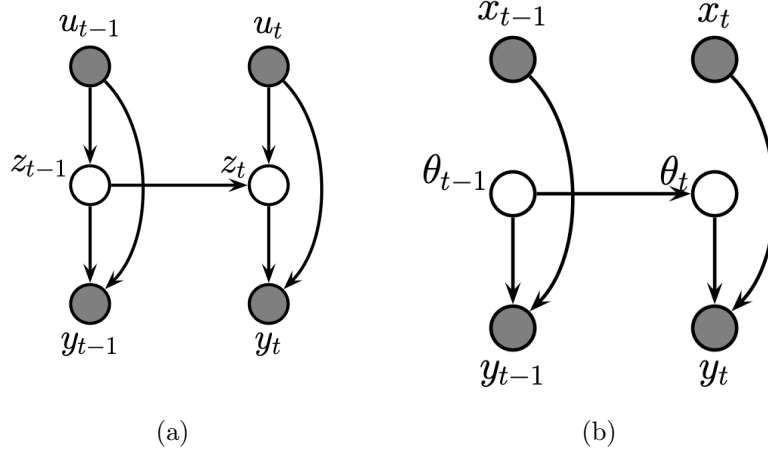


Figure 18.1: State Space Model (SSM) as a graphical model (shaded nodes are observed and unshaded are latent). (a) Standard SSM. (b) Specialized version for online parameter inference.

We start by defining a State Space Model (SSM), which is a way to define a joint distribution over a sequence of latent states z_t and observed outputs y_t , given a set of observed inputs or controls u_t . The model has the following form:

$$p(y_{1:T}, z_{1:T} | u_{1:T}) = p(z_1) p(y_1 | z_1, u_1) \prod_{t=2}^T p(y_t | z_t, u_t) p(z_t | z_{t-1}, u_t) \quad (18.4)$$

See Figure 18.1(a) for an illustration.

The distribution $p(z_t | z_{t-1}, u_t)$ is called the dynamics (or transition) model and specifies how the latent state changes over time. The distribution $p(y_t | z_t, u_t)$ is called the observation (or measurement) model and specifies the likelihood of the observations given the hidden state (and optionally the controls).

Given the above model, the online inference problem (also called Bayesian filtering) refers to recursively updating the belief state to give $p(z_t | y_{1:t}, u_{1:t})$, taking advantage of the Markov property of the SSM. We discuss algorithms for computing this in Section 18.3. In the offline setting, where the entire dataset is available, we may instead be interested in the Bayesian smoothing problem, which computes $p(z_{1:T} | y_{1:T}, u_{1:T})$ or corresponding marginals, such as $p(z_t | y_{1:T}, u_{1:T})$. However, this is beyond the scope of this chapter. (If you are interested in this case, see e.g., Courts et al. [336], Dowling et al. [448] for methods based on VI and Kamthe et al. [864] for a method based on Expectation Propagation.)

18.2.2 Examples

As a concrete example, consider the problem of tracking a moving object (such as an airplane) given noisy observations of its location. In this case, z_t represents the position and velocity of the object, y_t represent the observation (e.g., from a radar or camera), and u_t represents the input control signals chosen by the pilot which steer the plane. The belief state represents a distribution over the object's state given the noisy observations seen to date.

In the case of neural network parameter estimation, the latent state z_t corresponds to the parameters θ_t of the network, y_t represents the target label, and the input u_t represents the features or covariates x_t which are passed into the network. In this application, the hidden states (network parameters) are assumed to be unaffected by the inputs, so there is no edge from u_t to z_t in the graphical model. The resulting simplified model is shown in Figure 18.1(b). See Section 18.2.5 for more details.

18.2.3 Linear Dynamical Systems

We now consider an important special case of an SSM known as a Linear Dynamical System (LDS). This model is widely used because it supports exact inference, as we discuss in Section 18.3.2. In addition, it will be a building block for the approximate inference algorithms which we will consider later in this chapter.

In an LDS, both the transition model and dynamics model are conditional Gaussian distributions whose means are linear functions of their inputs, and whose variances are fixed. More precisely, we assume

$$p(z_t | z_{t-1}, u_t) = \mathcal{N}(z_t | A_t z_{t-1} + B_t u_t + a_t, Q_t) \quad (18.5)$$

$$p(y_t | z_t, u_t) = \mathcal{N}(y_t | C_t z_t + D_t u_t + c_t, R_t) \quad (18.6)$$

Here Q_t is the covariance of the system noise, and R_t is the covariance of the observation noise. If the process is stationary, then the model parameters $(A_t, B_t, C_t, D_t, a_t, c_t, Q_t, R_t)$ will be independent of t .

This formulation is known as the affine form of an LDS (see Särkkä and Svensson [1461, Sec 6.4]). We can remove a_t by appending a value of 1 to the input u_t and appending a_t as a column to B_t . Similarly we can remove c_t by absorbing it into D_t . However, sometimes it is convenient to explicitly include these bias (offset) terms, just like we do in neural networks. Alternatively, since the inputs are fixed, we can absorb $B_t u_t$ into a_t and $D_t u_t$ into c_t .

18.2.4 Nonlinear dynamical systems

The assumption that the SSM has a linear dynamics model and a linear observation model (Equations 18.5-18.6) is quite restrictive. However, it is easy to relax this by defining a Gaussian SSM where the mean is a nonlinear function of its input:

$$p(z_t | z_{t-1}, u_t) = \mathcal{N}(z_t | d(u_t, z_{t-1}), Q_t) \quad (18.7)$$

$$p(y_t | z_t, u_t) = \mathcal{N}(y_t | f(u_t, z_t), R_t) \quad (18.8)$$

Note that, since the noise is additive, we can rewrite the above model as follows:

$$z_t = d_t(z_{t-1}) + \epsilon_t \quad (18.9)$$

$$y_t = f_t(z_t) + \epsilon'_t \quad (18.10)$$

where $\epsilon_t \sim \mathcal{N}(0, Q_t)$ is the system noise, $\epsilon'_t \sim \mathcal{N}(0, R_t)$ is the observation noise, and we have defined the shorthands $d_t(z_{t-1}) = d(u_t, z_{t-1})$ and $f_t(z_t) = f(u_t, z_t)$.

18.2.5 SSMs for online Bayesian deep learning

In this section, we discuss how to formulate the problem of online neural network parameter estimation as inference in an SSM, where $z_t = \theta_t$ are the unknown network parameters, $u_t = x_t$ are the inputs to the network, and y_t are the observed outputs.

Observation model

The observation model f corresponds to the neural network that maps inputs to outputs given the hidden parameters, so $f_t(\theta_t) = f(x_t, \theta_t)$. The assumption that the output distribution is Gaussian is often reasonable for real-valued targets (although we may want to use more robust loss functions to handle outliers, as discussed in Duran-Martin et al. [467], Knoblauch et al. [904]). In this case, we often assume $R_t = \sigma_r^2 I$, where $\sigma_r > 0$ is a hyper-parameter.

However, when the targets are discrete labels or integer counts, we should replace the Gaussian observation distribution by a more suitable distribution from the exponential family. In this case, we assume $f_t(\theta_t)$ represents the mean parameter of the output distribution, and $h_t(\theta_t)$ represents the natural parameter.¹ For example, consider neural network classification. In this case, $f(x_t, \theta_t)$ represents the probability of each class label, and $h(x_t, \theta_t)$ represents the logits. We then have

$$p(y_t | \theta_t, x_t) = \text{Cat}(y_t | f(x_t, \theta_t)) = \text{Cat}(y_t | S(h(x_t, \theta_t))) \quad (18.11)$$

where S is the softmax function.

¹Note that in Jones et al. [849], they use f for the natural parameters and h for the mean parameters, which is the opposite of our convention.

Dynamics model

It is common to assume (e.g., Kurle et al. [951], Titsias et al. [1617]) that the dynamics model is an Ornstein-Uhlenbeck process, i.e.,

$$p(\theta_t | \theta_{t-1}) = \mathcal{N}(\theta_t | d_t(\theta_{t-1}), Q_t) \quad (18.12)$$

$$d_t(\theta_{t-1}) = \gamma_t \theta_{t-1} + (1 - \gamma_t) \mu_0 \quad (18.13)$$

$$Q_t = (1 - \gamma_t^2) \Sigma_0 \quad (18.14)$$

where $0 \leq \gamma_t \leq 1$ is the degree of drift and $\mathcal{N}(\theta_0 | \mu_0, \Sigma_0)$ is the prior. If $\gamma_t = 1$, then the dynamics is the identity function and there is no noise ($Q_t = 0$), so the latents are constant, i.e., $p(\theta_t | \theta_{t-1}) = \delta(\theta_t - \theta_{t-1})$.

The $(1 - \gamma_t^2)$ factor in Equation 18.14 implies a process with stationary marginals $p(\theta_t) = \mathcal{N}(\theta_t | \mu_0, \Sigma_0)$. Another common choice is to use a “variance exploding” Gaussian random walk, where $\gamma_t = 1$ but $Q_t = \sigma_q^2 I$ for $\sigma_q > 0$. This can provide a form of regularisation. In this case, the quantity σ_q^2 / σ_r^2 , which is the ratio of the measurement precision to the dynamics precision, controls the learning rate of the system. (This can be seen by looking at the form of the Kalman gain matrix in Equation 18.25; see Ollivier [1258] for more discussion.) We discuss how to choose these hyper-parameters in Section 18.3.6.

18.3 Online Inference

We now discuss how to perform sequential (online) inference in SSMs using some common deterministic algorithms. (For a more detailed account, see e.g., Särkkä and Svensson [1461], Triantafyllopoulos [1634].)

18.3.1 Bayesian filtering

If we define $\mathcal{D}_t = (u_t, y_t)$, we can recursively estimate the belief state $p(z_t | \mathcal{D}_{1:t}) = p(z_t | u_t, y_t, \mathcal{D}_{1:t-1})$, by using a slight generalisation of Equation 18.1, as follows:

$$\underbrace{p(z_t | \mathcal{D}_{1:t-1}, u_t)}_{\text{posterior predictive}} = \int \underbrace{p(z_t | z_{t-1}, u_t)}_{\text{dynamics}} \underbrace{p(z_{t-1} | \mathcal{D}_{1:t-1})}_{\text{previous posterior}} dz_{t-1} \quad (18.15)$$

$$\underbrace{p(z_t | \mathcal{D}_{1:t})}_{\text{new posterior}} = \frac{1}{Z_t} \underbrace{p(y_t | z_t, u_t)}_{\text{likelihood}} \underbrace{p(z_t | \mathcal{D}_{1:t-1}, u_t)}_{\text{prior}} \quad (18.16)$$

$$Z_t = \underbrace{p(y_t | \mathcal{D}_{1:t-1}, u_t)}_{\text{observation predictive}} \quad (18.17)$$

Equation 18.15 is called the predict step. This uses the Chapman-Kolmogorov equation to compute the one-step-ahead predictive distribution for the latent state, before seeing the outcome y_t ; this becomes the prior for step t . Equation 18.16 is called the update step. This uses Bayes’ rule to compute the posterior after seeing the outcome y_t . Finally, Equation 18.17 is the one-step-ahead predictive distribution for observations evaluated at y_t .²

18.3.2 Kalman filtering

In this section, we present the celebrated Kalman filter, which applies the Bayesian filtering equations from Section 18.3 to the LDS model defined in Section 18.2.3. The result is a recursive procedure for estimating the belief state $p(z_t | y_{1:t}, u_{1:t}) = \mathcal{N}(z_t | \mu_t, \Sigma_t)$. The derivation is well-known and can be found in many textbooks (see, e.g., Särkkä and Svensson [1461, Sec 6.3] or Murphy [1213, Sec 8.2.2]). For brevity, we drop the conditioning on the inputs u_t and absorb them into the bias terms.

The predict step is as follows:

$$p(z_t | y_{1:t-1}) = \mathcal{N}(z_t | \mu_{t|t-1}, \Sigma_{t|t-1}) \quad (18.18)$$

$$\mu_{t|t-1} = A_t \mu_{t-1} + a_t \quad (18.19)$$

$$\Sigma_{t|t-1} = A_t \Sigma_{t-1} A_t^\top + Q_t \quad (18.20)$$

²We call $p(y_t | u_t, \mathcal{D}_{1:t-1})$ the observation predictive, to distinguish it from $p(z_t | u_t, \mathcal{D}_{1:t-1})$, which is the posterior predictive. The former can be computed without the latter, as emphasized by the martingale posterior methodology of Holmes and Walker [773].

The predict step takes $O(P^3)$ time (recall that P is the dimension of the latent state).

The update step is as follows:

$$p(z_t \mid y_{1:t}) = \mathcal{N}(z_t \mid \mu_t, \Sigma_t) \quad (18.21)$$

$$\hat{y}_t = C_t \mu_{t|t-1} + c_t \quad (18.22)$$

$$e_t = y_t - \hat{y}_t \quad (18.23)$$

$$S_t = C_t \Sigma_{t|t-1} C_t^\top + R_t \quad (18.24)$$

$$K_t = \Sigma_{t|t-1} C_t^\top S_t^{-1} \quad (18.25)$$

$$\mu_t = \mu_{t|t-1} + K_t e_t \quad (18.26)$$

$$\Sigma_t = \Sigma_{t|t-1} - K_t S_t K_t^\top = \Sigma_{t|t-1} - K_t C_t \Sigma_{t|t-1} \quad (18.27)$$

Here e_t is the innovation or residual error, S_t is the innovation covariance, and K_t is the Kalman gain matrix. The update step takes $O(P^2C + PC^2 + C^3)$ time (recall that C is the dimension of the observation).

We see that the posterior mean is the prior predicted mean plus the gain matrix times the residual, and the posterior covariance is the prior covariance minus a positive definite term, indicating that the uncertainty is usually reduced after conditioning. This becomes clearer when we use the information form, where we work with precision matrices:

$$\Sigma_t^{-1} = \Sigma_{t|t-1}^{-1} + C_t^\top R_t^{-1} C_t \quad (18.28)$$

We see that the posterior precision is the prior precision plus a term which depends on the precision of the observations.

Finally, we note that the observation predictive is given by

$$p(y_t \mid y_{1:t-1}) = \mathcal{N}(y_t \mid \hat{y}_t, S_t) \quad (18.29)$$

from which we can compute Z_t .

18.3.3 Extended Kalman filtering

We now extend the Kalman filtering equations to the case of a nonlinear (but still Gaussian) SSM. We do this by linearising both the dynamics and measurement models around the current prior mean, and then performing a step of Kalman filtering. This is known as the Extended Kalman Filter (EKF), and was first developed at NASA [1524].

To explain the method in more detail, consider a nonlinear function $g(z)$. We can compute a linear approximation of this around some point μ using a first-order Taylor series expansion:

$$\hat{g}(z) = g(\mu) + J(z - \mu) \quad (18.30)$$

where $J = \text{Jac}(g)|_\mu$ is the Jacobian of g evaluated at μ (see also Section 8.1). If $g : \mathbb{R}^D \rightarrow \mathbb{R}^C$, then computing J costs $O(C|g|)$ using reverse mode auto-differentiation, or $O(D|g|)$ using forward mode, where $|g|$ is the cost of evaluating g . We can now approximate the original conditional Gaussian distribution, with nonlinear mean, by a conditional Gaussian whose mean is an affine function of its input:

$$\mathcal{N}(g(z), \Sigma) \approx \mathcal{N}(Az + b, \hat{\Sigma}) \quad (18.31)$$

$$A = \text{Jac}(g)|_\mu \quad (18.32)$$

$$b = g(\mu) - A\mu \quad (18.33)$$

$$\hat{\Sigma} = \Sigma \quad (18.34)$$

Applying this to our SSM, we get the following:

$$\hat{d}_t(z_{t-1}) = \mu_{t|t-1} + A_t(z_{t-1} - \mu_{t-1}) \quad (18.35)$$

$$\mu_{t|t-1} = d_t(\mu_{t-1}) \quad (18.36)$$

$$A_t = \text{Jac}(d_t)(\mu_{t-1}) \quad (18.37)$$

$$\hat{f}_t(z_t) = \hat{y}_t + C_t(z_t - \mu_{t|t-1}) \quad (18.38)$$

$$\hat{y}_t = f_t(\mu_{t|t-1}) \quad (18.39)$$

$$C_t = \text{Jac}(f_t)(\mu_{t|t-1}) \quad (18.40)$$

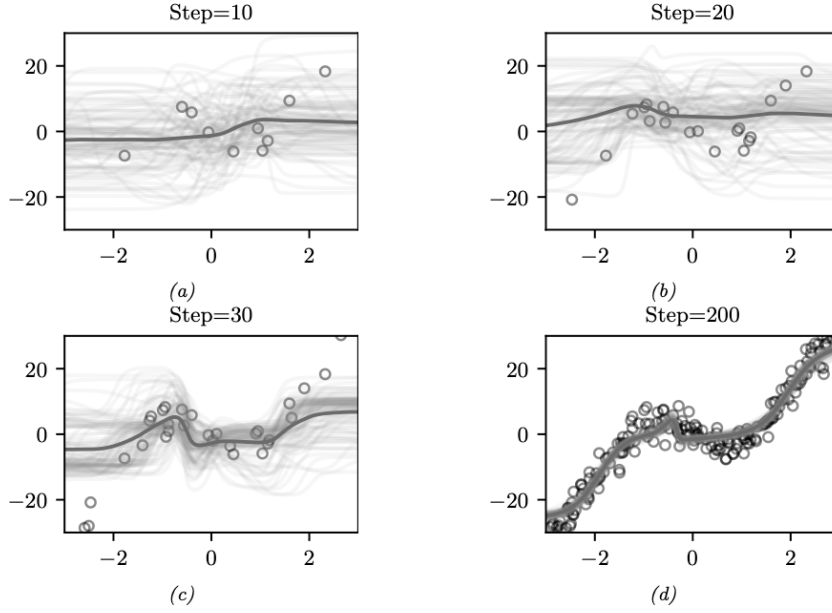


Figure 18.2: EKF applied to an MLP for a nonlinear 1d regression problem. We show samples from the posterior predictive distribution over the output space after seeing the first 10, 20, 30 and 200 observations. (For a video of this, see https://github.com/probml/probml-data/blob/main/data/ekf_mlp_demo.mp4.) The code to reconstruct this plot (written by Gerardo Duran-Martin) is at https://github.com/probml/JSL/blob/main/jsl/demos/ekf_mlp.py.

We then apply one step of Kalman filtering.

The EKF can be iterated, by relinearising the model around the updated posterior mean, instead of the prior mean. This is equivalent to a Gauss-Newton optimisation [111] and can improve the accuracy of the approximation.

The EKF can be applied to estimate the parameters of a neural network regression model as discussed in Puskorius and Feldkamp [1348, 1349], Singhal and Wu [1518]. See Figure 18.2 for an example. However, this takes $O(P^3)$ time per step, in general, which is intractable if the number of parameters is large. In addition, the method is restricted to Gaussian observations. In Section 18.4, we will present faster and more general methods.

18.3.4 Statistical Linear Regression

We now present an alternative way to linearise a nonlinear Gaussian system, known as Statistical Linear Regression (SLR) [1461], which can give higher accuracy but at increased computational cost. The basic idea is to compute the parameters of the linear Gaussian approximation by minimising an expected regression loss:

$$\mathcal{N}(g(z), \Sigma) \approx \mathcal{N}(Az + b, \hat{\Sigma}) \quad (18.41)$$

$$A, b = \arg \min \mathbb{E}_{y,z} [\|y - Az - b\|_2^2] = \arg \min \mathbb{E}_z [\|g(z) - Az - b\|_2^2] \quad (18.42)$$

$$A = \text{Cov}[y, z] \mathbb{V}[z]^{-1} \quad (18.43)$$

$$b = \mathbb{E}[y] - A\mathbb{E}[z] \quad (18.44)$$

$$\hat{\Sigma} = \mathbb{V}[y] - A\mathbb{V}[z]A^\top \quad (18.45)$$

(A similar method can be used for the dynamics model, substituting z_{t-1} and z_t for z and y .) The integrals needed to evaluate the relevant expectations and variances can be approximated using Gaussian quadrature. This gives rise to a variety of popular algorithms, including the Unscented KF [857] (also called sigma-point filtering), the Gauss-Hermite KF [42], and the cubature KF [41]. We can also iterate these methods, giving rise to the posterior linearisation filter [577].

These techniques have been applied to neural network parameter estimation (e.g., de Lima et al. [387]), but they are even less scalable than EKF-based methods. We therefore now turn to VI methods, which are much faster.

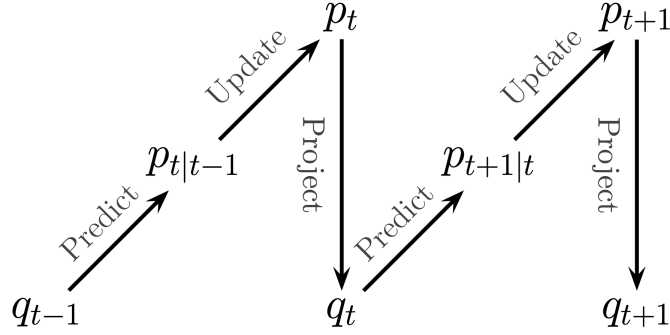


Figure 18.3: Illustration of the predict-update-project cycle for the measurement step, where $p_t = p_{t|t}$. We omit the corresponding predict-project step for the dynamics, which computes $q_{t|t-1} \approx p_{t|t-1}$, since this is often easy to compute.

18.3.5 Variational Inference

The methods discussed in Sections 18.3.3 and 18.3.4 for computing an approximate Gaussian posterior do not give any guarantees that the result is the optimal Gaussian approximation. We therefore turn to variational inference methods, where we explicitly solve for the optimal Gaussian approximation.

Predict step

In the predict step, we can approximate the integral in the Chapman-Kolmogorov equation 18.15 by minimising the forwards KL:

$$q_{t|t-1}(z_t) = \arg \min_{q \in \mathcal{Q}} D_{\text{KL}}(p_{t|t-1}(z_t) \| q(z_t)) \quad (18.46)$$

This can be solved by moment matching, as is well known, giving rise to

$$q_{t|t-1}(z_t) = \mathcal{N}(z_t | \mu_{t|t-1}, \Sigma_{t|t-1}) \quad (18.47)$$

$$\mu_{t|t-1} = \mathbb{E}_{p_{t|t-1}}[z_t] \quad (18.48)$$

$$\Sigma_{t|t-1} = \text{Cov}_{p_{t|t-1}}[z_t] \quad (18.49)$$

In the case of nonlinear dynamics, we can use linearisation (as in the EKF discussed in Section 18.3.3) or Gaussian quadrature methods (as in the UKF discussed in Section 18.3.4) to approximate these moments. In the case of BDL with static parameters, where the dynamics is the identity function and the system noise is $Q_t = \sigma_q^2 I$, the predict step is much simpler, and reduces to $\mu_{t|t-1} = \mu_{t-1}$ and $\Sigma_{t|t-1} = \Sigma_{t-1} + \sigma_q^2 I$.

Update step

In the update step, a natural approach is again to minimize the forwards KL:

$$q_t(z_t) = \arg \min_{q \in \mathcal{Q}} D_{\text{KL}}(p_{t|t}(z_t) \| q(z_t)) \quad (18.50)$$

$$p_{t|t}(z_t) = \frac{1}{Z_t} p(y_t | z_t) q_{t|t-1}(z_t) \quad (18.51)$$

This can be interpreted as taking the prior distribution, $q_{t|t-1}$, updating it using Bayes' rule, and then projecting the resulting posterior $p_{t|t}$ back to the space of tractable distributions $\mathcal{Q}$, as illustrated in Figure 18.3. This is known as Assumed Density Filtering [1152] or Minimum Divergence Filtering [372]. In the offline (smoothing) setting, this corresponds to the Expectation Propagation algorithm [1181] (see also Chapter 25). As in the predict step, minimising $D_{\text{KL}}(p \| q)$ amounts to matching the moments of the two distributions.³

³There is an interesting connection between ADF and the Gaussian Filtering (GF) methods discussed in Section 18.3.3 and Section 18.3.4. As noted in Wüthrich et al. [1783], we can interpret GF as first computing an approximate jointly Gaussian distribution $q(z_t, \tilde{y}_t)$ by minimising $D_{\text{KL}}(p(z_t, \tilde{y}_t | y_{1:t-1}) \| q)$ using moment matching, and then conditioning this joint on the event $\tilde{y}_t = y_t$, which corresponds to the standard Gaussian updating procedure, to get an approximation $q_t(z_t | y_{1:t})$. By contrast, ADF first computes the (locally) exact posterior $p_t(z_t | y_{1:t})$ and then approximates it (by moment matching) to compute $q_t(z_t | y_{1:t})$. Thus ADF is more accurate but also much slower.

The ADF methodology has been applied to online neural network training, for example the expectation backpropagation algorithm of Soudry et al. [1545], the probabilistic backpropagation algorithm of Hernandez-Lobato and Adams [743] and the work of Ghosh et al. [621]. However, in high-dimensional spaces, ADF is intractable, since it first requires computing the exact one-step posterior $p_{t|t}$ before extracting its moments (which in turn requires computing high dimensional integrals).

We therefore turn to the more common VI method, in which we minimize the reverse KL, corresponding to information projection (resulting in mode seeking behaviour, rather than mode covering). This is more tractable than ADF and forwards KL since the expectations are wrt the chosen distribution q rather than wrt the true posterior p . More precisely, we compute

$$q_t(z_t) = \arg \min_q D_{\text{KL}}(q(z_t) \| p_{t|t}(z_t)) \quad (18.52)$$

$$= \arg \min_q \mathbb{E}_q \left[\log \frac{q(z_t)}{p(y_t|z_t)q_{t|t-1}(z_t)/Z_t} \right] \quad (18.53)$$

$$= \arg \min_q \underbrace{D_{\text{KL}}(q(z_t) \| q_{t|t-1}(z_t)) - \mathbb{E}_q[\log p(y_t|z_t)]}_{\mathcal{L}_{t|t}(\mu_t, \Sigma_t)} \quad (18.54)$$

where we dropped the constant Z_t from the last line since it does not depend on q and hence does not affect the arg min. The resulting objective takes the familiar form of an expected negative log likelihood (NLL) loss plus a KL regulariser to the prior, and corresponds to an online version of the negative ELBO (evidence lower bound).

It can be shown (e.g., Cao et al. [231], Jones et al. [849], Lambert et al. [965], Zeno et al. [1835]) that the fixed points of the objective $\mathcal{L}_{t|t}$ satisfy the following conditions:

$$\mu_t = \mu_{t|t-1} - \Sigma_{t|t-1} \mathbb{E}_{\mathcal{N}(z_t|\mu_t, \Sigma_t)} [\nabla_{z_t} \ell(z_t, y_t)] \quad (18.55)$$

$$\Sigma_t^{-1} = \Sigma_{t|t-1}^{-1} + \mathbb{E}_{\mathcal{N}(z_t|\mu_t, \Sigma_t)} [\nabla_{z_t}^2 \ell(z_t, y_t)] \quad (18.56)$$

where $\ell(z_t, y_t) = -\log p(y_t|z_t)$ is the NLL loss.

Note that the unknown parameters μ_t, Σ_t occur on both the left and right hand sides of the above equations, forming an implicit update. Therefore, in general, we must iterate these equations to find a solution. We discuss suitable algorithms of this kind in Section 18.4. However, in the case of linear Gaussian systems, one can show that the expectations on the RHS have analytic solutions, so we can perform the update in one step, using the Kalman filtering equations discussed in Section 18.3.2.

18.3.6 State estimation vs parameter estimation

So far, we have assumed the parameters representing the dynamics model and the observation model are known. More precisely, we have been computing $p(z_t | \mathcal{D}_{1:t}, \phi_t)$, where ϕ_t are the parameters of the dynamics function d_t and the observation function f_t . This problem is called “state estimation”. If the parameters are unknown, we can add them to the hidden state space and perform joint state and parameter estimation, by computing $p(z_t, \phi_t | \mathcal{D}_{1:t})$. This is known as “system identification” and is a much more complicated problem (see e.g., Courts et al. [335], Dowling et al. [448]).

Fortunately, in the case of BDL, the dynamics and observation models typically have very few free parameters (e.g., $\phi_t = (\gamma_t, \sigma_q)$ for OU dynamics in Section 18.2.5), since the unknown network parameters θ_t can be treated as hidden states z_t . We can therefore treat ϕ as hyper-parameters, which can be estimated using empirical Bayes (i.e., by optimising them wrt the marginal likelihood). For example, Titsias et al. [1617] optimise the drift term γ_t of the OU dynamics by maximizing the one-step-ahead log predictive likelihood, $p(y_t | y_{1:t-1}, \gamma_t)$, which can be computed from the normalization constant Z_t of the approximate posterior distribution. By optimising the drift term γ_t online, they can adapt to non-stationarity in the data.

18.4 Bayesian Online Natural Gradient

In this section, we present the Bayesian Online Natural Gradient (BONG) algorithm of Jones et al. [849], which minimizes the VI objective in Equation 18.54 using natural gradient descent. BONG is a version of the Bayesian learning rule (BLR) of Khan and Rue [883] (see Chapter 15) adapted to the online case. The BLR update also works by natural gradient descent on a negative ELBO, but it uses a batch objective rather than the recursive one in Equation 18.54. Thus BONG differs from BLR by incorporating the online inference tools presented in

Sections 18.2-18.3, namely dynamics models, iterative priors, and filtering-style updates with a sequential predict-update structure.⁴ In addition, we discuss how to compute the various expectations required by these updates in an efficient manner. The main advantages of BONG are that it defines a computationally efficient update for Bayesian neural networks during online learning, and it provides a unified framework encompassing many existing methods.

18.4.1 Background

We first introduce some background material. We assume q_t is in the exponential family, where ψ_t are the natural parameters, and ρ_t are the dual (moment) parameters. For a Gaussian, the natural and moment parameters are given by

$$\psi_t = (\Sigma_t^{-1} \mu_t, -\frac{1}{2} \Sigma_t^{-1}) \quad (18.57)$$

$$\rho_t = (\mu_t, \mu_t \mu_t^\top + \Sigma_t) \quad (18.58)$$

It can be shown that the Jacobian of the mapping from dual parameters to natural parameters for any exponential-family distribution is equal to the Fisher information matrix:

$$F_{\psi_t} = \frac{\partial \rho}{\partial \psi} \Big|_{\psi_t} \quad (18.59)$$

Consequently, the natural gradient wrt the natural parameters is equal to the regular gradient wrt the dual parameters, i.e.

$$F_{\psi_t}^{-1} \nabla_{\psi_t} \ell(\psi) = \nabla_{\rho_t} \ell(\rho) \quad (18.60)$$

Converting from the natural to the dual parameters has the same computational cost as computing the Fisher information matrix.

We continue to write the latent state as z_t and the input or controls as u_t , to reflect that our treatment applies to general state estimation problems. As a reminder, for online neural network parameter estimation we take $z_t = \theta_t$ as the unknown network parameters and $u_t = x_t$ as the inputs to the network.

18.4.2 Update step

We can minimize Equation 18.54, or maximize its negative, by performing iterative updates following the natural gradient direction. At iteration i , the update becomes

$$\psi_t^i = \psi_t^{i-1} + \alpha_t F_{\psi_t^{i-1}}^{-1} \nabla_{\psi_t^{i-1}} \left(\mathbb{E}_{\psi_t^{i-1}} \left[\log p(y_t | f_t(z_t)) - \text{D}_{\text{KL}}(q_{\psi_t^{i-1}} \| q_{\psi_{t|t-1}}) \right] \right) \quad (18.61)$$

$$= \psi_t^{i-1} + \alpha_t \nabla_{\rho_t^{i-1}} \left(\mathbb{E}_{\psi_t^{i-1}} \left[\log p(y_t | f_t(z_t)) - \text{D}_{\text{KL}}(q_{\psi_t^{i-1}} \| q_{\psi_{t|t-1}}) \right] \right) \quad (18.62)$$

where $\alpha_t > 0$ is a learning rate. (The Bayes By Backprop (BBB) method of Blundell et al. [174] uses a similar update, but uses standard gradient for ψ_t , instead of natural gradient. This requires more iterations due to the lack of preconditioning.)

If we initialise with $\psi_t^0 = \psi_{t|t-1}$, and just do a single iteration, the gradient of the KL regulariser vanishes, and we just perform a gradient update on the expected log likelihood, as follows:

$$\psi_t = \psi_{t|t-1} + \nabla_{\rho_{t|t-1}} \underbrace{\mathbb{E}_{\psi_{t|t-1}} [\log p(y_t | f_t(z_t))]}_{\bar{\mathcal{L}}_{t|t-1}} \quad (18.63)$$

This is the approach used in the BONG method of Jones et al. [849]. Even though we have dropped the KL term, we are using an implicit (algorithmic) form of regularisation by initialising q_t at the prior and not moving far from it (since we just take one step). Furthermore, it can be shown that a single gradient step with unit learning rate ($\alpha_t = 1$) yields the exact Bayesian update in the conjugate case.

⁴When using a full covariance posterior, we will show that BONG takes $O(P^2)$ time per step whereas BLR takes $O(P^3)$. Both methods can be reduced to $O(P)$ time per step in the case of diagonal covariance, but BONG is still a constant factor faster, and is fully deterministic.

In Jones et al. [849], they show that the result of this explicit update is given by the following:

$$\Sigma_t^{-1} = \Sigma_{t|t-1}^{-1} - \underbrace{\mathbb{E}_{\mathcal{N}(z_t|\mu_{t|t-1}, \Sigma_{t|t-1})}[\nabla_{z_t}^2 \log p(y_t|f_t(z_t))]}_{G_t} \quad (18.64)$$

$$\mu_t = \mu_{t|t-1} + \Sigma_t \underbrace{\mathbb{E}_{\mathcal{N}(z_t|\mu_{t|t-1}, \Sigma_{t|t-1})}[\nabla_{z_t} \log p(y_t|f_t(z_t))]}_{g_t} \quad (18.65)$$

Algorithm 22: BONG algorithm, assuming static parameters

```

for  $t = 1$  to  $\infty$  do
  // Predict
   $\mu_{t|t-1} \leftarrow \mu_{t-1}$ ,  $\Sigma_{t|t-1} \leftarrow \Sigma_{t-1}$ 
  // Update
  define loss  $\ell_t(z_t) \leftarrow -\log p(y_t | u_t, z_t)$ 
  compute  $g_t$  and  $G_t$  using methods in Section 18.4.3 applied to  $\ell_t$ 
  compute  $\Sigma_t^{-1}$  using Equation 18.64
  compute  $\mu_t$  using Equation 18.65

```

See Algorithm 22 for the resulting pseudocode. In Section 18.4.3, we explain how to compute the g_t and G_t terms, and in Section 18.4.4, we discuss the computational complexity of the algorithm. The bottom line is that, if we use a full covariance Gaussian, the method takes either $O(P^2)$ or $O(P^3)$ time, depending on algorithmic choices, which is intractable for most applications. Therefore, in Section 18.4.5 we consider a diagonal approximation, which only takes $O(P)$ time. Finally, in Section 18.4.6, we consider using a low rank representation, which is more expressive than diagonal but still scales linearly in P .

18.4.3 Computing the expectations

In Cao et al. [231] they approximate the relevant expectations using Gaussian quadrature, which they combine with natural gradient descent to get a method they call “NANO” (short for natural gradient Gaussian approximation filter). However, quadrature does not scale to high dimensions.

Another approach (used e.g., in the RVGA method of Lambert et al. [965]) is to use a Monte Carlo approximation, where we sample $z_t^m \sim \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$ and then compute

$$g_t^{\text{MC}} = \frac{1}{M} \sum_{m=1}^M g_t^m, \quad g_t^m = \nabla_{z_t^m} \log p(y_t|f_t(z_t)) \quad (18.66)$$

$$G_t^{\text{MC}} = \frac{1}{M} \sum_{m=1}^M G_t^m, \quad G_t^m = \nabla_{z_t^m}^2 \log p(y_t|f_t(z_t)) \quad (18.67)$$

Since computing the Hessian is intractable in high dimensions, we can replace G_t^{MC} with an empirical Fisher (EF) approximation (e.g., Martens [1135]), defined as follows:

$$G_t^{\text{MCEF}} = -\frac{1}{M} G_t^{1:M} (G_t^{1:M})^\top, \quad G_t^{1:M} = [g_t^1, \dots, g_t^M] \quad (18.68)$$

Another approach is to linearise the observation model, as in the EKF (Section 18.3.3). In addition, we approximate the observation distribution with its first two (conditional) moments, following Ollivier [1258], Tronarp et al. [1637]. That is, we use the following approximate likelihood:

$$p_t^{\text{LG}}(y_t|z_t) = \mathcal{N}(y_t|\hat{f}_t(z_t), R_t) \quad (18.69)$$

$$\hat{f}_t(z_t) = \hat{y}_t + C_t(z_t - \mu_{t|t-1}) \quad (18.70)$$

$$\hat{y}_t = f_t(\mu_{t|t-1}) \quad (18.71)$$

$$C_t = \text{Jac}(f_t)|_{\mu_{t|t-1}} \quad (18.72)$$

$$R_t = \mathbb{V}[y_t|z_t = \mu_{t|t-1}] \quad (18.73)$$

This yields

$$g_t^{\text{Lin}} = C_t^\top R_t^{-1}(y_t - \hat{y}_t), \quad G_t^{\text{LinHess}} = -C_t^\top R_t^{-1} C_t \quad (18.74)$$

To avoid having to compute the Jacobian (which takes $O(C|f|)$ time, and hence can be slow for high-dimensional targets), we can use an empirical Fisher style approximation:

$$g_t^{\text{LinEF}} = g_t^{\text{Lin}}, \quad G_t^{\text{LinEF}} = -g_t^{\text{Lin}}(g_t^{\text{Lin}})^\top \quad (18.75)$$

Finally, we can approximate the expectations by using a plugin or delta approximation, where we replace $\mathcal{N}(z_t | \mu_{t|t-1}, \Sigma_{t|t-1})$ with $\delta(z_t - \mu_{t|t-1})$. This gives rise to

$$g_t^\delta = \nabla_{z_t} \log p(y_t | f_t(z_t))|_{\mu_{t|t-1}}, \quad G_t^\delta = \nabla_{z_t}^2 \log p(y_t | f_t(z_t))|_{\mu_{t|t-1}}, \quad (18.76)$$

We can also combine the plugin approximation with the linearisation or empirical fisher approximation to get cheaper approximations to G_t :⁵

$$G_t^{\delta\text{Lin}} = G_t^{\text{LinHess}}, \quad G_t^{\delta\text{EF}} = -g_t^\delta (g_t^\delta)^\top \quad (18.77)$$

Thus in summary, we have identified 4 ways to approximate g_t : quadrature, sampling, linearisation, and plugin. Similarly, we have identified 7 ways to approximate G_t : quadrature, sampling (with or without EF), linearisation (with or without EF), and plugin (with or without EF). The different methods trade off speed vs accuracy, as we discuss in Section 18.4.4.

18.4.4 Computational complexity

In this section, we briefly discuss the computational complexity (running time) of the above algorithms. The time depends on three factors: (1) the form of the variational posterior q , (2) the method chosen for estimating g_t and G_t , and (3) whether we do a one step update on the implicitly regularised expected log likelihood (ELL) objective $\bar{\mathcal{L}}_{t|t-1}$ defined in Equation 18.63, or do multiple update steps on the explicitly regularised ELBO objective $\mathcal{L}_{t|t}$ defined in Equation 18.54. The details can be found in Jones et al. [849], but we briefly summarize some key results here.

Suppose we use a full covariance q , we compute g_t and G_t by linearising the model, as in Equation 18.74, and we do a one-step update on the ELL objective. One can show that the resulting update is equivalent to the information form of the Kalman filter shown in Equation 18.28:

$$\Sigma_t^{-1} = \Sigma_{t|t-1}^{-1} - G_t = \Sigma_{t|t-1}^{-1} + C_t^\top R_t^{-1} C_t \quad (18.78)$$

$$\mu_t = \mu_{t|t-1} + \Sigma_t G_t = \mu_{t|t-1} + \Sigma_t C_t^\top R_t^{-1} (y_t - \hat{y}_t) \quad (18.79)$$

To avoid having to convert from Σ_t^{-1} to Σ_t , we can use the matrix inversion lemma, which gives the standard form of the KF, which we repeat here for convenience:

$$K_t = \Sigma_{t|t-1} C_t^\top (R_t + C_t \Sigma_{t|t-1} C_t^\top)^{-1} \quad (18.80)$$

$$\mu_t = \mu_{t|t-1} + K_t (y_t - \hat{y}_t) \quad (18.81)$$

$$\Sigma_t = \Sigma_{t|t-1} - K_t C_t \Sigma_{t|t-1} \quad (18.82)$$

This method, which is equivalent to the CM-EKF method of Ollivier [1258], Tronarp et al. [1637], takes $O(CP^2)$ time.

If we approximate G_t using the Monte Carlo EF approximation in Equation 18.68, as is done in the RVGA method of Lambert et al. [965], the time is $O(MP^2)$. If we approximate G_t using the linearised EF approximation in Equation 18.75, we can reduce the time to $O(P^2)$. All other variants (including the original BLR method of Khan and Rue [883]) take $O(P^3)$ time. Therefore, in the sections below, we discuss more restricted parameterisations for q , so we can reduce the complexity to linear in P .

⁵To define $G_t^{\delta\text{Lin}}$, we linearise the natural parameters of the observation distribution $h_t(z_t)$ instead of the mean parameters $f_t(z_t)$ as in Equation 18.69. See Jones et al. [849] for details and the proof that $G_t^{\delta\text{Lin}} = G_t^{\text{LinHess}}$.

18.4.5 Diagonal case

The easiest way to speed up the algorithm is to assume that the covariance matrix of the variational posterior is diagonal, i.e., we use $\psi_t = (\sigma_t^{-2}\mu_t, -\frac{1}{2}\sigma_t^{-2})$, or equivalently $\rho_t = (\mu_t, \text{diag}(\mu_t^2 + \sigma_t^2))$, where the product and exponents are taken elementwise. The relevant update equations can be found in Jones et al. [849, Sec E.3-E.4], but all can be done in $O(P)$ time.

When combined with the linearised Hessian approximation of Equation 18.74, and applied to the one-step ELL objective, we recover the variational diagonal EKF method of Chang et al. [258]. When combined with the MC Hessian approximation of Equation 18.67, and applied to the multi-step ELBO objective, we recover an online version of the VON algorithm from Khan et al. [881]. If instead we use the MC EF approximation of Equation 18.68, we recover an online version of the VOGN algorithm from Khan et al. [881]. (See also the recently improved VON method in Shen et al. [1510].) If we use standard gradient descent (instead of NGD), and optimise the mean parameters ρ_t instead of ψ_t , we recover an online version of the BBB method of Blundell et al. [174].

18.4.6 Low-rank case

Finally, we consider the case where the posterior precision matrix is diagonal plus low rank (DLR), i.e., $\Sigma_t^{-1} = D_t + W_t W_t^\top$, where D_t is a diagonal matrix of size $P \times P$, and W_t is a tall and skinny matrix of size $P \times R$, where $R \ll P$ is the rank of the approximation. This representation captures correlation among the latent parameters, unlike the diagonal representation, yet still scales linearly in the dimension, as we will see. (It is also possible to use the DLR approximation for the posterior covariance, Σ_t , as explained in Tomczak et al. [1622].)

Directly applying NGD to the ELL objective of Equation 18.63 or the ELBO objective of Equation 18.54 is challenging, because the Fisher information matrix for the DLR parameterisation cannot be easily inverted. We therefore first derive the update for the full covariance natural parameters, leveraging the fact that the prior $\Sigma_{t|t-1}^{-1}$ is DLR to make this efficient, and then use SVD to project the resulting posterior precision matrix back to low-rank form. This can be done in $O(R^2 P)$ time; for the details, see Jones et al. [849, Sec E.5].

If we combine this method with the linearised Hessian approximation for G_t given in Equation 18.74, and apply to the one-step ELL objective, we recover the LOFI (low-rank filtering) method of Chang et al. [259]. If instead we use the MC-EF approximation for G_t given in Equation 18.68⁶ and apply to the iterated ELBO objective, we recover an online variant of the SLANG method of Mishkin et al. [1186]. (SLANG uses a randomized SVD algorithm from Halko et al. [702], whereas LOFI uses a deterministic SVD method.)

An alternative to the above methods is to use standard gradient descent to optimise the DLR parameterisation. This avoids the need to compute the Fisher information matrix. This simple scheme (which is called “BBB-DLR” in Jones et al. [849]) has not been experimentally evaluated, as far as we know.

We can also drop the diagonal term from the DLR representation, giving rise to a degenerate low-rank matrix. This is the basic idea behind the low-rank Kalman filter (LRKF) and variants described in Duran-Martin et al. [468]. In more detail, LRKF uses the approximation $\Sigma_t = W_t W_t^\top$. Note that the resulting belief state, $p(z_t|y_{1:t})$ is not a proper Gaussian distribution, since the covariance matrix is singular, but it is still a valid probability measure. Also the corresponding predictive distribution over observations, $p(y_{t+1}|y_{1:t})$, is generally a proper Gaussian. Although the lack of a diagonal term means we need to use a larger rank to capture a desired statistical performance, empirically the algorithm still offers a better compute-accuracy tradeoff than LOFI, which relies on the DLR representation.

18.4.7 Connections to second-order optimisation

In this section, we show an equivalence between the BONG algorithm — when combined with certain approximations and applied to the case of inference for static parameters — and standard second-order optimisation algorithms. In particular, consider the BONG update for Σ_t given in Equation 18.64 and μ_t given in Equation 18.65. For static parameters, we have $\mu_{t|t-1} = \mu_{t-1}$ and $\Sigma_{t|t-1} = \Sigma_{t-1}$, since there is no added noise. If we combine this with the delta-function (plugin) approximation to the expectations, we get g_t^δ from Equation 18.76 for the gradient, and $G_t^{\delta\text{EF}}$ from Equation 18.77 for the Hessian (assuming we also use the EF approximation).

⁶Note that, to implement the MC-EF method, we have to be able to sample from the DLR prior, without inverting the full matrix. An efficient algorithm for this, based on EM, is given in the LRVGA paper of Lambert et al. [966]. This takes $O(R(R+M)P)$ time to draw M samples.

The resulting update becomes

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + g_t g_t^\top \quad (18.83)$$

$$\mu_t = \mu_{t-1} + \Sigma_t g_t \quad (18.84)$$

The mean update is equivalent to a standard preconditioned gradient ascent update. Furthermore, the precision update is a cumulative sum of approximate Hessians, as in most second-order optimisers. In fact, if we use $\Sigma_t^{\frac{1}{2}}$ as the preconditioner in Equation 18.84, and add a learning rate, we exactly recover full-matrix Adagrad [458]. Most adaptive gradient methods (including Adagrad [458], Shampoo [691], and Adam [894]) use a square root preconditioner, but performance can be comparable or improved without it [1048]. There are also very close connections between this (delta function) approximation to BONG and mirror descent methods used in online learning (e.g., Chérif-Abdellatif et al. [289], Lyu and Tsang [1095], van der Hoeven et al. [1656]). More connections between VI and popular gradient based learning methods used in the deep learning community are discussed in Khan and Rue [883] (see Chapter 15).

18.5 Conclusions

This chapter has shown how we can learn the parameters of a deep neural network online using tools from Bayesian filtering. Incorporating an SSM for parameter dynamics allows the model to maintain plasticity and is especially well-suited for nonstationary or continual learning tasks. The resulting learning rules have a two-step structure: a predict step to advance the dynamics and an update step to assimilate the next observation into the belief state. linearising the network (and dynamics) makes these steps analytically tractable and allows us to use classic Kalman filter methods. More generally we can use variational inference for the update step to maintain a belief state within a given parametric family. In the Gaussian case, optimising the online variational objective is efficiently approximated by one step of natural gradient descent, formalised in the BONG update. Making the update computationally efficient leads to a family of methods indexed by how the relevant integrals and Hessians are approximated (including Monte Carlo, quadrature, linearisation, plugin, empirical Fisher) and how the posterior covariance is represented (diagonal, low rank, DLR, or full covariance). Open questions for future research include how to determine the optimal tradeoff between computational efficiency and model expressiveness or fidelity for a given problem; richer dynamic models or learning the dynamics for specific domains; a deeper understanding of the connection to second-order optimisers; and whether variational families other than Gaussians yield effective methods for certain problems.

Part IV

Ensemble methods

Chapter 19

Introduction and connections with approximate Bayesian inference

Deep ensembles [962] train the same neural-network architecture several times from different random initialisations and combine the resulting predictive distributions. Their appeal lies in their simplicity, straightforward parallelisation, competitive predictive performance, and broad applicability.

Deep ensembles were initially presented as a non-Bayesian alternative to approximate Bayesian inference methods such as variational inference (VI; Chapter 12) [962, 1274]. Subsequent conceptual and empirical work has shown that, in several studied settings and under realistic computational budgets, their predictive distributions can be closer to a high-fidelity Bayesian reference than those of standard single-basin approximations [826, 1765]. This does not make a standard deep ensemble an exact posterior approximation: its members are independently optimised solutions with equal weights, rather than posterior draws or basins weighted by posterior mass.

This chapter introduces the deep-ensemble procedure (Section 19.1), examines its connection with Bayesian model averaging (Section 19.2), discusses ensemble-based developments (Section 19.3), and considers the method under fixed computational budgets (Section 19.4).

19.1 Deep ensembling procedure

Let w denote the network parameters, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote the training data, and let J denote the number of ensemble members. Deep ensembling proceeds as follows:

1. For each $j = 1, \dots, J$, initialise the network independently and use stochastic gradient descent (SGD), or another optimiser, to obtain a local solution w_j . When the training objective is, up to an additive or multiplicative constant, the negative log-posterior $-\log p(w \mid \mathcal{D})$, these local solutions can be interpreted as local maximum a posteriori (MAP) estimates.
2. Form the equally weighted ensemble predictive distribution

$$\hat{p}_{\text{DE}}(y \mid x, \mathcal{D}) := \frac{1}{J} \sum_{j=1}^J p(y \mid x, w_j). \quad (19.1)$$

The original formulation trains probabilistic neural networks using proper scoring rules and optionally uses adversarially perturbed examples to encourage locally smoother predictive distributions [962]. For Gaussian-likelihood regression, member j commonly predicts

$$p(y \mid x, w_j) = \mathcal{N}(y \mid \mu_j(x), \sigma_j^2(x)). \quad (19.2)$$

Although the resulting equal mixture is generally not itself Gaussian, its first two moments are

$$\begin{aligned}\bar{\mu}(x) &= \frac{1}{J} \sum_{j=1}^J \mu_j(x), \\ v_{\text{DE}}(x) &= \frac{1}{J} \sum_{j=1}^J \sigma_j^2(x) + \frac{1}{J} \sum_{j=1}^J (\mu_j(x) - \bar{\mu}(x))^2 \\ &= \frac{1}{J} \sum_{j=1}^J (\sigma_j^2(x) + \mu_j^2(x)) - \bar{\mu}(x)^2.\end{aligned}\tag{19.3}$$

The first contribution to $v_{\text{DE}}(x)$ averages the per-model aleatoric variance, while the second measures between-model disagreement.

The procedure is simple but computationally expensive: training, storing, and evaluating J full models generally costs approximately J times as much as using one model. Because the training runs are independent, they can be distributed across devices, which reduces wall-clock time when sufficient parallel hardware is available but does not remove the total computation or storage cost.

19.2 Deep ensembles as approximate Bayesian inference

The Bayesian posterior predictive distribution, or Bayesian model average (BMA), is

$$p(y \mid x, \mathcal{D}) = \int p(y \mid x, w) p(w \mid \mathcal{D}) dw.\tag{19.4}$$

There are two useful perspectives on the relationship between deep ensembles and approximate Bayesian inference. First, an average of the form in Equation (19.1) is an ordinary Monte Carlo estimator of Equation (19.4) only when the w_j are samples from $p(w \mid \mathcal{D})$. Deep-ensemble members are instead local solutions obtained by optimisation from different initialisations, so their empirical distribution need not match the posterior in its locations, relative mass assigned to different basins, or within-basin spread [692].

The second perspective asks how a small number of evaluations of $p(y \mid x, w)$ should be allocated under a fixed computational budget. If predictive distributions vary less within a basin than across functionally distinct basins, representing several basins may capture more of the predictive variation relevant to Equation (19.4) than allocating all evaluations to one basin [1765]. This is a heuristic for finite-budget approximation, not a claim that independently optimised modes are superior to well-mixed exact posterior samples. Figures 19.1 and 19.2 illustrate this distinction.

This comparison also appears in image classification. Using full-batch Hamiltonian Monte Carlo (HMC) as a high-fidelity reference, Izmailov et al. [826] found that deep ensembles were substantially closer to the HMC predictive distribution than mean-field VI and standard SGD, and were competitive with several stochastic-gradient MCMC methods (Figure 19.3).

These results support a qualified revision of the usual description of deep ensembles. Rather than being dismissed as wholly unrelated to Bayesian inference, they can provide a useful finite-budget approximation to Bayesian predictive marginalisation and, in some settings, a closer approximation to a high-fidelity Bayesian reference than several traditional single-basin methods [826, 1765]. Their empirical success also motivates methods that combine coverage of multiple basins with explicit within-basin marginalisation, as discussed in Section 19.3.

The Bayesian interpretation should not be read literally, however. Standard deep ensembles assign equal weight to independently trained members rather than weighting basins by posterior probability or evidence. They do not sample from $p(w \mid \mathcal{D})$ and ordinarily represent between-member variation without explicitly representing uncertainty within each basin. It is therefore more accurate to describe Equation (19.1) as an equally weighted finite predictive mixture that may approximate the BMA than as a posterior-weighted model average.

It is also useful to distinguish Bayesian model averaging from model combination. As discussed by Minka [1182], BMA represents uncertainty among hypotheses in a specified probabilistic model using posterior weights, whereas model combination may be used to construct a richer predictive family. The members of a deep ensemble use the same neural-network architecture, but their predictive mixture may nevertheless represent distributions that no single member represents. This distinction is another reason not to identify a standard deep ensemble directly with the exact BMA. The same caution applies to MC Dropout [566]: its stochastic mask distribution defines an approximation rather than posterior sampling, so properties of the exact posterior cannot be assumed automatically.

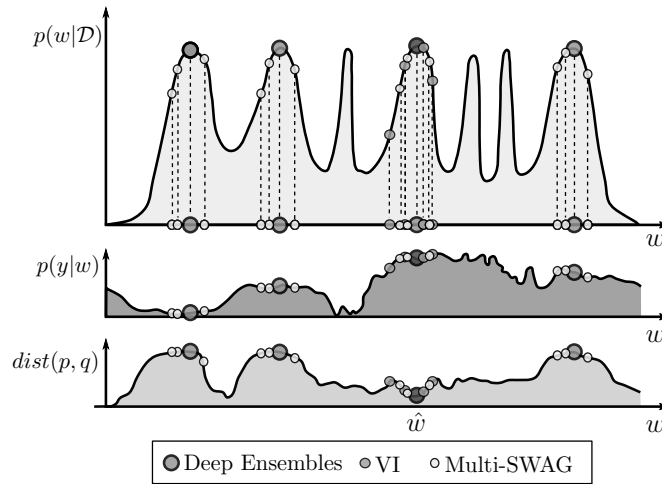


Figure 19.1: Conceptual illustration of finite approximations to the BMA $p(y | x, \mathcal{D}) = \int p(y | x, w)p(w | \mathcal{D}) dw$. Top: a multimodal parameter posterior together with three forms of approximation. The large circular markers represent deep-ensemble point estimates; the compact group of small markers around one mode represents a unimodal VI approximation; and the small markers distributed around several modes represent MultiSWAG, a mixture of local Gaussian approximations. Dashed vertical lines indicate the corresponding parameter-space evaluation locations. Middle: the predictive distribution $p(y | x, w)$ as a function of w for one test input x ; in this illustration, predictions change less within a mode than between modes. Bottom: the discrepancy between the BMA and a finite approximation as a function of the location of one additional evaluation, given that one mode is already represented. Under the assumptions illustrated here, representing a previously unexplored basin reduces the discrepancy more than adding another evaluation within an already represented basin. Figure adapted from Wilson and Izmailov [1765].

Several works analyse different aspects of deep ensembles as approximate Bayesian inference [310, 368, 453, 718, 1306, 1765].

A central lesson is that practical Bayesian inference in deep learning is almost always approximate, so comparisons should state both the target distribution and the computational budget. Exact posterior draws, when available and representative of the posterior, are valid Monte Carlo inputs to the BMA in Equation (19.4). The practical question is how efficiently a method obtains evaluations that represent the predictive variation relevant to that integral. Deep ensembles can outperform less expensive single-basin approximations at fixed computational budgets, but this empirical observation should not be interpreted as a claim that deep ensembles are superior to well-mixed exact posterior sampling.

19.3 Developments of deep ensembles

A single Laplace or mean-field variational approximation (Chapter 13) usually represents uncertainty within one local basin. Deep ensembles instead represent several basins using one optimised point in each. These complementary properties motivate mixture approximations that perform explicit marginalisation within several basins.

Stochastic Weight Averaging–Gaussian (SWAG) [1116] fits a local low-rank-plus-diagonal Gaussian approximation using an SGD trajectory. MultiSWAG [1765] repeats this construction in several independently discovered basins and combines the resulting Gaussian components into a mixture approximation.

The approximate-inference competition reported by Wilson et al. [1770] received 337 submissions from 12 teams, and the highest-ranked methods all used multimodal ensemble constructions. The winning team used a uniformly weighted mixture of diagonal-Gaussian approximations, each fitted with improved Variational Online-Newton (iVON) [1047]. One of the shared second-place teams replaced the standard optimisation trajectories in MultiSWAG with stochastic-gradient Langevin dynamics (SGLD), while another used a computationally efficient sequential anchored-ensemble construction guided by MCMC [1306].

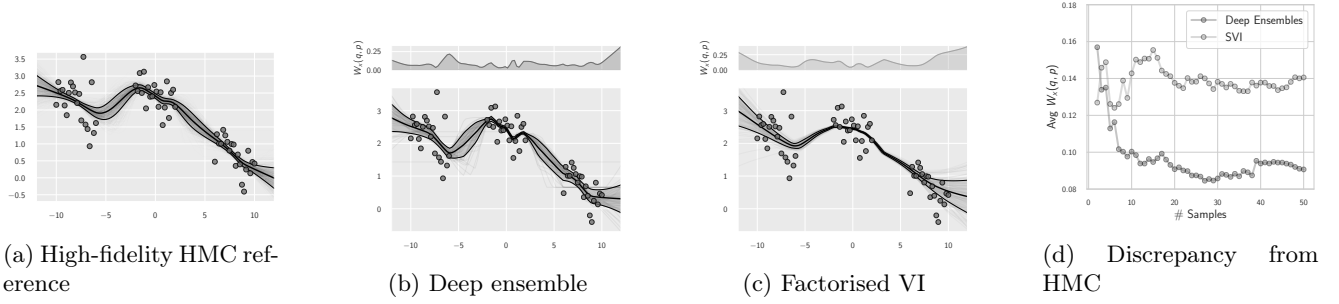


Figure 19.2: Approximating a high-fidelity HMC reference predictive distribution on a one-dimensional regression task. **(a)** Reference predictive distribution obtained by combining 10 long HMC chains. **(b)** Deep-ensemble predictive distribution using 50 independently trained networks. **(c)** Predictive distribution from factorised variational inference. In panels **(b)** and **(c)**, the narrow upper strip reports the input-wise Wasserstein discrepancy from the HMC reference, while the lower plot shows the corresponding predictive distribution. **(d)** Average Wasserstein distance from the HMC reference as a function of the number of predictive components or samples. On this example, the multi-basin deep ensemble approaches the HMC reference more closely than the single-basin VI approximation, which is overconfident between data clusters. Figure adapted from Wilson and Izmailov [1765].

19.4 Are deep ensembles necessary?

Deep ensembles perform well in an absolute sense, but training, storing, and evaluating J full models is expensive. SWAG [1116] generally uses one main training trajectory followed by a posterior-collection phase, although drawing J predictive samples still requires J model evaluations. Deep kernel learning (Chapter 24) [1769] requires one forward pass through its feature extractor for each test input, followed by GP prediction in the learned feature space. A standard deep ensemble instead requires approximately J independent training runs and J forward passes at prediction time.

Several alternatives reduce the cost of obtaining diverse predictions, including BatchEnsemble [1739], snapshot ensembles [795], fast geometric ensembling [581], rank-1 Bayesian neural networks [474], and anchored ensembles [1306]. These methods make different trade-offs between diversity, memory, training cost, and fidelity to a Bayesian predictive distribution.

Any fair comparison should control for computational budget and for the property being evaluated. Retraining and averaging J copies of one network may outperform one training run of that network, but a single larger model may use the same computation more effectively when predictive accuracy is the only objective [3].

Predictive accuracy is not the only relevant objective, however. Deep-ensemble disagreement provides an operational measure of epistemic uncertainty that can support tasks such as exploration in Bayesian optimisation [78]. Simply increasing the size of a deterministic model does not, by itself, provide such uncertainty. An explicit uncertainty mechanism is required, such as an ensemble, a variational posterior (Chapter 13), a GP head (Chapter 24), or a Laplace approximation. Comparisons between ensembles and single models should therefore distinguish predictive accuracy from the quality and computational cost of uncertainty estimation.

19.5 Summary

Deep ensembles form an equally weighted predictive mixture from independently trained neural networks. They are not posterior samples or posterior-mass-weighted basin averages, but they can provide an effective finite-budget approximation to Bayesian predictive marginalisation when their members represent functionally distinct solutions. Their combination of simplicity, parallelisability, predictive performance, and useful disagreement estimates explains their practical success, while their linear training, storage, and prediction costs motivate more efficient and more explicitly Bayesian multi-basin approximations.

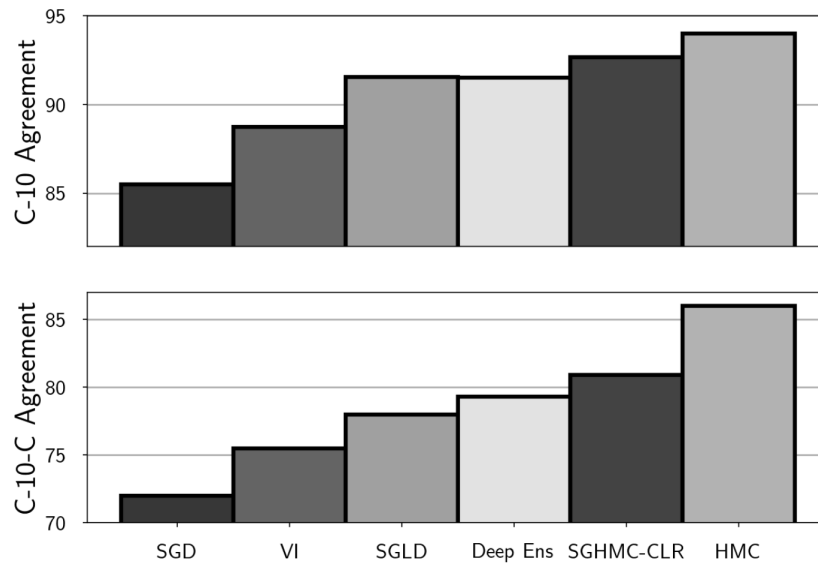


Figure 19.3: Agreement with a high-fidelity HMC reference on image classification. The upper panel reports agreement on clean CIFAR-10, while the lower panel reports agreement on CIFAR-10-C averaged across corruption types and intensities. The bars compare standard SGD, mean-field VI (labelled “VI” in the plot), stochastic-gradient Langevin dynamics (SGLD), deep ensembles, stochastic-gradient HMC with a cyclical learning-rate schedule (SGHMC-CLR), and an HMC reference. For the HMC reference, one HMC chain is compared with the ensemble of two other chains. Higher agreement indicates greater similarity of the top-one predictions. Deep ensembles are substantially closer to HMC than VI and are competitive with SGLD, while SGHMC-CLR is closer to HMC in both panels. All methods use ResNet-20-FRN trained on CIFAR-10. Figure adapted from Izmailov et al. [826].

Chapter 20

Hybrid ensemble sampling methods

This chapter highlights that the effectiveness of navigating high-dimensional, non-convex, and multimodal posterior landscapes typical in Bayesian deep learning can be significantly enhanced by hybrid approaches that combine sampling with optimization-based techniques, or ensembles thereof.

20.1 On the Efficacy of Optimization-Informed Initialization

The posterior landscapes of interest are typically non-convex and characterized by complex geometries, often featuring multiple, disjoint modal regions of significant probability mass. The successful navigation of this terrain under finite computational constraints necessitates a strategy for balancing global exploration against local exploitation.

To formalize the problem of posterior approximation via sampling, it is instructive to decompose the total approximation error. Following the framework presented in Sommer et al. [1535], this error may be attributed to three distinct sources:

1. **Initialization Error:** An error component induced by commencing the sampling trajectory at a point of low probability, distant from the posterior’s *typical set*—the high-probability volume where the bulk of the posterior mass is concentrated [144].
2. **Discretization Error:** An error component arising from the numerical approximation of a continuous-time stochastic differential equation (e.g., the Langevin diffusion) by a discrete-time iterative algorithm.
3. **Monte Carlo (MC) Error:** A statistical error component due to the use of a finite ensemble of samples to estimate expectations under the true posterior. This error is inversely related to the number of effective samples drawn.

In this chapter, we are primarily concerned with the mitigation of initialization and Monte Carlo errors, as their interplay motivates the adoption of hybrid sampling strategies. Adjusting the discretization error can be done independently of this interplay.

Most MCMC samplers are designed to be ergodic processes in order to explore the full support of the target distribution. This exploratory capacity distinguishes them from optimization algorithms. Consider, for example, the relationship between Stochastic Gradient Descent and Stochastic Gradient Langevin Dynamics (SGLD; see Chapter 2). With potential $U(\theta) := -\log p(\theta \mid \mathcal{D})$, SGD performs gradient descent $\theta_{t+1} = \theta_t - \eta_t \nabla U(\theta_t)$. The unadjusted Langevin algorithm and its stochastic variant (SGLD) add isotropic Gaussian perturbations,

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{2} \nabla U(\theta_t) + \sqrt{\eta_t} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (20.1)$$

Equivalently,

$$\theta_{t+1} = \theta_t + \frac{\eta_t}{2} \nabla \log p(\theta_t \mid \mathcal{D}) + \boldsymbol{\zeta}_t, \quad \boldsymbol{\zeta}_t \sim \mathcal{N}(\mathbf{0}, \eta_t \mathbf{I}). \quad (20.2)$$

The injected noise facilitates transitions between modes, yet when far from the typical set it can slow the approach to high-density regions [1097, 1737].

Optimization procedures such as SGD [809], while typically not suited to sample from the posterior, are highly adept at rapidly identifying points of high posterior density, such as Maximum A Posteriori (MAP) estimates [1108]. This observation motivates a synergistic, two-stage methodology that integrates optimization and sampling [1533]. This hybrid approach may be conceptualized as follows:

1. **Optimization-based Initialization:** An optimization algorithm is first employed to find a parameter vector θ^* that resides within a high-density region of the posterior, effectively minimizing an objective such as the unnormalized negative log-posterior or the negative log-likelihood. This stage is designed to substantially mitigate the initialization error by positioning the sampler’s starting point within or proximal to the typical set. For most relevant prior choices (having an unbounded support like an isotropic Gaussian), the obtained θ^* lies in a region of reasonably high prior density, making it a valid initialization for the MCMC procedure.
2. **Localized Posterior Sampling:** Subsequent to optimization, an MCMC sampler is initialized at θ^* . Relieved of the protracted and often inefficient burn-in phase (induced by the potentially very complex neural network posterior landscape) required when starting from a random point in the parameter space, the sampler can allocate its computational budget to the more focused task of characterizing the local posterior geometry, thereby more efficiently reducing the Monte Carlo error.

In other words, the practical imperative for such a hybrid strategy is to overcome the inefficiency and instability of samplers initialized from the prior, $p_0(\theta)$, particularly in high dimensions [1534]. As shown in Table 20.1, prior-initialized chains frequently fail to discover parameter configurations that yield meaningful predictive performance. This phenomenon is also demonstrated in Figure 20.1 for a simple overparameterized linear network $f(x) = w_1 w_2 x$ with known multimodal posterior, and maximum posterior density at $w_1 = w_2 = 1$ and $w_1 = w_2 = -1$. A sampler with an optimized initialization rapidly converges to and explores one of the true posterior modes, whereas a prior-initialized counterpart fails to explore higher density regions as well as the initialized one within the same budget of sampling steps. This gain in initial sampling efficacy is particularly important for high-dimensional models. Moreover, the computational overhead of the preliminary optimization phase is often negligible relative to the total cost of sampling, yet yields substantial improvements in the overall efficiency of posterior exploration and characterization [1534–1536]. Even in the very small-scale example provided in Figure 20.1, the first optimization stage only took 5% of the total runtime.

20.2 Parallelized Exploration for Mode Discovery

The hybrid two-stage strategy effectively solves the problem of finding an initial point close to the typical set. However, a significant challenge remains in posteriors characterized by multiple, well-separated modes (see Chapter 33 for related geometric and topological considerations). In high-dimensional spaces, the regions of low probability density that lie between these modes can practically form energy barriers for an MCMC sampler. Consequently, a single MCMC chain, even when run for a very long time, will likely remain confined to the basin of attraction of its initial mode.

This phenomenon is illustrated in Figure 20.1. The long-run simulation (bottom centre plot) is initialized in the posterior mode in the upper-right quadrant but never traverses the low-probability region around the origin to explore the second, symmetric mode in the lower-left quadrant. Relying on the formal ergodicity of a single chain to explore a multimodal landscape is, therefore, not a practically feasible strategy given that computation is usually constrained.

A simple and powerful solution is to leverage parallel computation by running multiple independent chains [600, 1421], a strategy shown in the lower right plot of Figure 20.1. The core idea is to initialize M different chains from distinct starting points $\{\theta_1^*, \dots, \theta_M^*\}$. If these starting points end up in different basins of attraction, the combined collection of samples from all M chains will form a much more faithful representation of the global posterior. This approach not only provides a mechanism for exploring functionally distinct modes but also represents a more efficient path to reducing the overall Monte Carlo error compared to running a single, exceptionally long chain. This approach is particularly appealing if parallel computing resources are available.

20.3 Variations of Hybrid Sampling

The conceptual framework of initializing sampling with optimization can be realized through several distinct methodologies, ranging from classical techniques to modern ensemble-based approaches. These methods differ in their approaches to managing the transition from optimization to sampling and in how they address the fundamental challenge of multimodality.

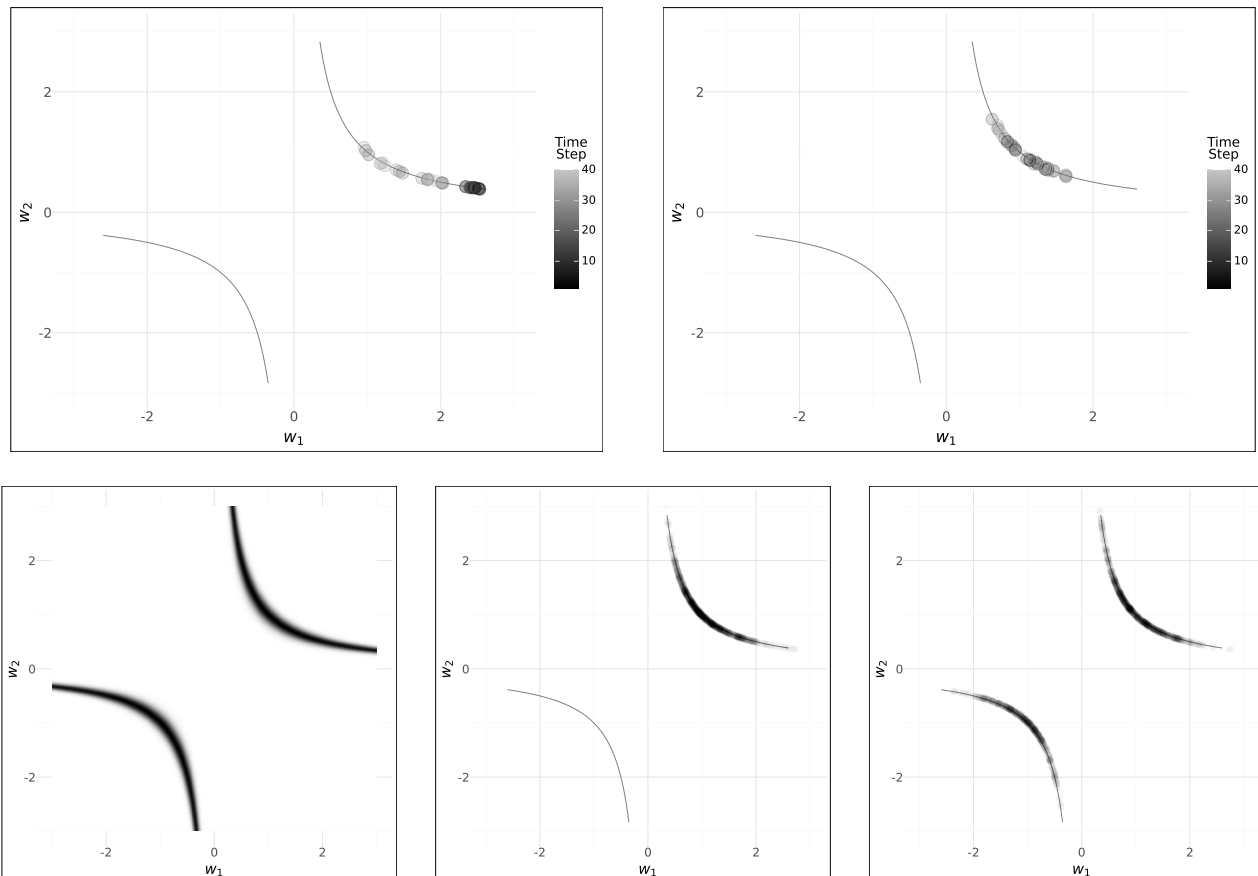


Figure 20.1: Upper left: Prior initialized sampling using the No U-Turn Sampler [NUTS; 762]. Upper right: Optimization initialized sampling using NUTS. Bottom left: Ground truth posterior obtained via numerical integration. Bottom centre: Long run sampling of a single NUTS chain from an optimized initialization. Bottom right: Samples obtained from multi-chain NUTS sampling with optimized initializations. For reference, the symmetry manifold $w_1 w_2 = 1$ is highlighted as a solid black line.

Table 20.1: The proportion of chains resulting in better root-mean-square error (RMSE) performance than a linear model for two UCI benchmark datasets [1638, 1813] (the higher the better). NUTS used 5,000 warmup samples. The Bayesian neural network used is a single hidden-layer Multi-Layer Perceptron with 50 hidden neurons. Values are averaged across 12 chains and 3 replications, and SD denotes standard deviation. Adapted from Table 5 of Sommer et al. [1534].

ACTIVATION	DATA SET	PROP. RMSE < LINEAR MODEL	SD
ReLU	CONCRETE	0.14	0.13
	ENERGY	0.31	0.13
SiLU	CONCRETE	0.14	0.10
	ENERGY	0.28	0.05
TANH	CONCRETE	0.58	0.25
	ENERGY	0.67	0.08

20.3.1 MAP Initialization and Tempered Transitions

A common practice in the broader Bayesian literature is to initialize a single MCMC chain from a previously computed MAP estimate [1671]. This represents the most direct instantiation of the two-stage hybrid approach: a “hard” transition from a pure optimization objective to a pure sampling objective.

A more gradual transition can be achieved by sampling from a **tempered target distribution**, a technique that smoothly interpolates between optimization and sampling. Following Chen et al. [273], Duffield et al. [461], Welling and Teh [1737], we can define a tempered posterior indexed by a temperature $T \in (0, \infty)$ as

$$p_T(\theta|\mathcal{D}) \propto [p(\theta|\mathcal{D})]^{1/T}. \quad (20.3)$$

The temperature T controls the “flatness” of the distribution. In the low-temperature limit $T \rightarrow 0$, the distribution p_T concentrates on the global maximizers of the original posterior density, while the corresponding Langevin dynamics approach gradient-based optimization. Conversely, setting $T = 1$ recovers the original, untempered posterior distribution.

This provides a formal mechanism for annealing. As an example, we can modify the SGLD update rule by keeping the posterior drift fixed and scaling the injected Langevin noise by the temperature. For $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, this gives

$$\theta_{t+1} = \theta_t + \frac{\eta_t}{2} \left(\nabla \log p_0(\theta_t) + \frac{N}{N_{\text{batch}}} \sum_{i \in \mathcal{B}_t} \nabla \log p(y_i | x_i, \theta_t) \right) + \zeta_t, \quad \zeta_t \sim \mathcal{N}(\mathbf{0}, \eta_t T \mathbf{I}). \quad (20.4)$$

Here, N is the dataset size, $\mathcal{B}_t \subset \{1, \dots, N\}$ is a minibatch index set of size N_{batch} , and the minibatch likelihood term is an unbiased estimator of $\nabla \log p(\mathcal{D} | \theta_t)$. By starting with a small temperature $T \ll 1$ and gradually increasing it to $T = 1$ over a series of iterations, the algorithm transitions smoothly from an optimization-dominated phase that quickly finds a mode to a sampling phase that explores its local geometry. This perspective also helps to formalize the close relationship between optimization and sampling. For instance, the exploration phase of cyclical SGLD [1854], which aims to find new posterior modes, operates close to the zero-temperature limit during exploration phases, hence approaching optimization before resetting to $T = 1$ to sample.

20.3.2 Ensemble-Based Initialization for Multi-Modality

While a single optimized starting point can effectively initialize sampling within one posterior mode, it cannot resolve the broader challenge of exploring a multimodal landscape. As discussed previously, traversing the low-probability regions between modes is often infeasible under computational constraints. To approximate a posterior with functionally distinct solutions, one can leverage **Deep Ensembles** [DE; 962], as introduced in Chapter 19.

The independent training runs of a DE typically converge to different local minima, yielding a set of diverse parameter vectors $\{\theta_1^*, \dots, \theta_M^*\}$. This collection of solutions provides an ideal set of starting points for multi-chain MCMC. The hybrid process then becomes the following:

1. Train a DE to obtain M distinct, high-performing models. For this step, one can also leverage previous empirical findings in the deep learning community and use readily available pre-trained weight vectors.

2. Initialize M parallel MCMC chains, with each chain m starting from its corresponding ensemble member’s parameters, θ_m^* .

This DE-initialized sampling approach turns posterior exploration into a *post hoc* procedure that can be applied to already-trained models to enhance their uncertainty quantification. This method has been shown to effectively capture diverse posterior structures in various task settings [461, 1534–1536] and is also referred to as *Bayesian Deep Ensembles (BDEs)*.

20.3.3 Related and Emerging Techniques

The core idea of finding diverse, high-quality starting points has inspired several related methods, such as the following three:

- **SWA-based Initialization:** Stochastic Weight Averaging (SWA) is an optimization technique that averages weights from later epochs of an SGD trajectory, often finding wider, more generalizable minima [824, 1128]. Instead of averaging the points along the SGD trajectory as in Izmailov et al. [824], the late-epoch iterates can serve as a set of correlated but diverse initializations for MCMC chains, offering a cheaper alternative to training a full Deep Ensemble [1303].
- **Diffusion-based Localization:** Stochastic localization methods build on score-based learning by perturbing data samples, which in the setting of sampling-based inference would be the model weights, with noise instead of optimizing them and then progressively learning a denoiser [755, 1199, 1541]. For example one such recent work introduces Stochastic Localization via Iterative Posterior Sampling, the so called SLIPS sampler, which uses MCMC-estimated denoisers to approximate samples from complex target distributions, offering a promising new class of initialization techniques [665]. Currently, however, these diffusion-based localization techniques struggle to cope with the very high dimensional posteriors present in modern neural network architectures (see Chapter 50 for details on diffusion and score-based methods).
- **Symmetry-Informed Initialization:** We can also leverage the inherent symmetries in neural network architectures for initialization (see Chapters 32 and 33). For many networks, such as Multi-Layer Perceptrons, one can permute the neurons within a hidden layer without changing the function computed by the network. As shown in Kobialka et al. [908], one can take a *single* pre-trained model and apply a set of M random, function-preserving permutations to its weights. This generates a “pseudo-ensemble” of M parameter vectors that are numerically distinct but functionally identical. Using these permuted vectors to initialize MCMC chains still seems to provide enough diversity to generate a well-working BDE while being highly efficient, requiring only one trained model.

20.4 Limitations

While the ensemble-based hybrid methods discussed offer a powerful, practical solution to exploring multimodal posterior landscapes, they are not without their own set of challenges [1533]. These limitations define an active frontier of research in Bayesian deep learning.

20.4.1 Weighting across Modes

One of the most significant theoretical challenges is how to combine the samples from different chains exploring distinct posterior modes. When running M independent chains, let S_m denote the number of retained samples from chain m , and let $\theta_{m,j}$ denote the j -th retained parameter sample from this chain. A simple and common approach is to form the final predictive distribution by treating each chain as an equally weighted component in a mixture,

$$p(y^*|x^*, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^M \left(\frac{1}{S_m} \sum_{j=1}^{S_m} p(y^*|x^*, \theta_{m,j}) \right). \quad (20.5)$$

However, this implicitly assumes that each of the M discovered modes contributes equally to the total posterior probability mass. This is rarely the case, as the true “weight” of a mode is its marginal likelihood, which is notoriously difficult to approximate in high dimensions, and its intractability is the reason for using MCMC in the first place. This problem of weighting components in a Bayesian model average is a classic challenge. Proposed solutions include, for example, stacking [1811, 1812], but in practice, most researchers and practitioners still resort to an equal weighting.

20.4.2 Computational and Resource Costs

Despite their relative efficiency compared to naïve sampling, hybrid methods remain computationally intensive. The initial step of training a DE requires the resources to train M separate neural networks. Subsequently, the MCMC sampling phase for each of the M chains adds further computational overhead. This combined cost, while often justifiable by the superior quality of the resulting uncertainty estimates, can be prohibitive for very large models or resource-constrained environments. Furthermore, storing the parameters for M full models and their corresponding posterior samples demands considerable memory and increases inference costs.

20.4.3 Fidelity of Mode Discovery

The entire framework rests on the assumption that the initial optimization phase successfully identifies diverse and meaningful posterior modes. However, this is not guaranteed under finite budget constraints. While unlikely, the members of a DE might converge to the same basin of attraction or to functionally almost equivalent solutions, limiting the exploratory advantage of the multi-chain approach [1534]. Moreover, there is a subtle mismatch between the objectives of the hybrid procedure: the optimizer seeks points that maximize the likelihood (or posterior density), while the Bayesian objective is to integrate over the entire typical set [144]. This can lead to the optimizer overlooking broad, flat modes that contribute significantly to the total posterior mass in favour of sharper, narrower peaks in the loss landscape [544].

20.4.4 Challenges of Local Exploration

Even after successfully initializing a chain in the close vicinity of a major posterior mode, fully exploring the local region can be difficult. The local geometry itself can be pathological, containing complex ridges, narrow funnels, or areas of high curvature that challenge even sophisticated samplers like Hamiltonian Monte Carlo [HMC, see Chapter 1; 456, 1229]. This can result in the sampler exploring only a small sub-region of the mode, leading to an underestimation of the local posterior variance. To avoid this, one could tune sampler hyperparameters such as its step size or mass matrix for each of the M chains. This, however, further complicates the practical application of these methods and stresses the need for high-performance and auto-tuned samplers.

Chapter 21

Practical applications of deep ensembles

Deep ensembles are related to Bayesian model averaging: under certain conditions an ensemble of independently trained networks can approximate the posterior predictive distribution and thus capture epistemic uncertainty, while also demonstrating robust performance across classification, regression, and calibration benchmarks. By training multiple neural networks independently and aggregating their predictions, deep ensembles approximate Bayesian model averaging — a long-standing principle for capturing epistemic uncertainty and improving generalization. Unlike many explicitly Bayesian neural network approaches, they achieve these benefits without requiring complex posterior inference ² or specialized training procedures ¹⁵, making them both conceptually simple and computationally tractable. In this chapter, we examine how these properties translate to applied settings, highlighting concrete examples where deep ensembles enhance model reliability, interpretability, and decision-making under uncertainty. We also discuss how their strengths and limitations inform the design of scalable, uncertainty-aware systems.

21.1 Introduction

Deep ensembles have emerged as one of the most widely used techniques for uncertainty quantification in modern deep learning practice. The core idea is deceptively simple: independently train multiple neural networks with different random initializations and stochastic training paths, then aggregate their predictions at test time, typically via averaging. This approach is strongly motivated by the Bayesian paradigm, which advocates for posterior distributions instead of point estimates.

This simplicity, combined with strong empirical performance, has made deep ensembles an appealing off-the-shelf tool for practitioners. They routinely deliver improved predictive accuracy and well-calibrated uncertainty estimates across a variety of domains, including computer vision, natural language processing, and forecasting tasks. Several Bayesian deep learning libraries readily provide support for deep ensembles ⁴⁷.

Deep ensembles have been used in practical applications in a variety of settings. In this section we will explore a non exhaustive list of practical applications of deep ensembles. Specifically we will cover applications in LLMs, weather forecasting, healthcare and reinforcement learning. Some of these applications (like in LLMs) can be surprising to the reader. Others, such as healthcare seem more intuitive given the high stakes sometimes involved in medical decisions. Given the high computational cost of deep ensembles, this must be offset by significant performance gains to justify their usage. In general, we can discern several common themes making deep ensembles useful in practice:

- Scarcity of data, here deep ensembles can be used to provide meaningful uncertainty estimates that can be used to obtain informative data in an online fashion. Ensembles can also be used to learn well-performing models that avoid overfitting.
- Large data diversity, here ensembles can be used to create specialized deep neural networks for subdivisions of the data. At test time the ensemble members can be recombined in various ways, often by first identifying the data point subdivision and applying the corresponding specialized ensemble member.
- Very high cost of predictive mistakes. Ensembles can provide well calibrated uncertainty estimates that can be used to avoid costly mistakes made by deep learning algorithms. Ensembles also tend to perform well for the task of selective classification, where the model can abstain from making a prediction when it is not confident enough.

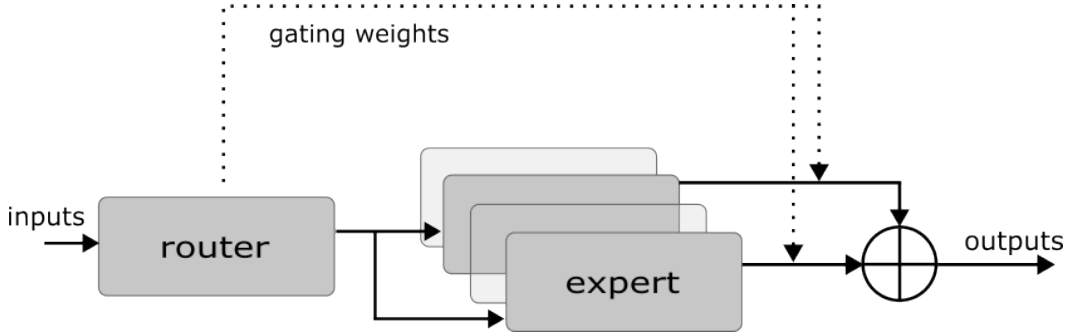


Figure 21.1: Mixture of Experts Layer. Each input vector is assigned to 2 of the 8 experts by a router. The layer’s output is the weighted sum of the outputs of the two selected experts. In Mixtral, an expert is a standard feedforward block as in a vanilla transformer architecture. Image and description from Jiang et al. [843].

Ensembles have also been proposed and evaluated without explicit reference to Bayesian inference [962, 1274]. This relaxes the Bayesian formalism and as we will see allows for practical methods, related to but separate from strict theory. Also, practical applications often introduce strict computational or other requirements. Thus, researchers might sometimes benefit from taking the above themes into account, when developing new ensembling methods.

21.2 Applications to Large Language Models

We start our discussion with practical applications of deep ensembles for Large Language Models (LLMs). At the time of writing of this text Large Language Models are the fastest growing field of deep neural networks. Ensembles are used in variety of sometimes overlooked ways in modern Large Language Models. They underpin several crucial parts of models in production providing significant efficiency gains in inference, easing generalization to new tasks and making finetuning more stable. In the next sections we will explore these different applications in detail.

21.2.1 Mixture-of-Experts

A popular variant of deep ensembles widely used in modern LLMs are Mixture of Experts models. In vanilla Deep Ensembles inference is performed on all ensemble members and the results are then averaged. Mixture of Experts models use a learnable router which selects a subset of ensemble members to be activated for every input. Conceptually, MoE implements a form of conditional or dynamic ensembling: the router composes a subset of experts per input, producing routing diversity that differs from static ensemble averaging. The result is a highly expressive model (given the high number of total parameters) that is also fast at inference time (due to the low number of activated parameters). Well known examples of Mixture of Experts models include Mixtral-of-Experts [843] and DeepSeekR1 [391–394].

We first reproduce a brief overview of the Mixtral Mixture of Experts layer (Figure 1). Given an input x , a router $G(x)$ and expert layers $E_i(x)$ with $i \in \{0, n-1\}$, the output of the MoE module for a given input x is determined by the weighted sum of the outputs of the expert networks, where the weights are given by the gating network’s output:

$$\sum_{i=0}^{n-1} G(x)_i \cdot E_i(x). \quad (21.1)$$

The crucial observation in this formulation is that, if the gating vector is sparse, one can avoid computing the outputs of experts whose gates are zero. This results in a model that is very flexible while being efficient at test time. The authors implemented the gating module by taking the softmax over the Top- K logits of a linear layer [1508]. Notably, more complicated methods have been explored in the literature [311, 717, 1875]. Concretely, the authors implement

$$G(x) := \text{Softmax}(\text{Top}_K(x \cdot W_g)) \quad (21.2)$$

where $(\text{Top}_K(\ell))_i := \ell_i$ if ℓ_i is among the Top_K coordinates of logits $\ell \in \mathbb{R}^n$ and $(\text{Top}_K(\ell))_i := -\infty$ otherwise. One can then modulate the amount of compute used to process each token, by appropriately choosing the value of K the number of experts used per token. In practice, the total number of experts n is kept much larger than the number of active experts K . For Mixtral the authors used the SwiGLU [843] architecture as the expert function

Model	Active Params	MMLU	HellaS
LLaMA 2 7B	7B	44.4%	77.1%
LLaMA 2 13B	13B	55.6%	80.7%
LLaMA 1 33B	33B	56.8%	83.7%
LLaMA 2 70B	70B	69.9%	85.4%
Mistral 7B	7B	62.5%	81.0%
Mixtral 8x7B	13B	70.6%	84.4%

Table 21.1: Comparison of Mixtral and LLaMA models on MMLU and HellaSwag benchmarks Jiang et al. [843].

$E_i(x)$. The number of total experts is $n = 8$ while the number of active experts is $k = 2$. Taking this all together, the output y for an input token x is computed as:

$$y = \sum_{i=0}^{n-1} \text{Softmax}(\text{Top}_2(x \cdot W_g))_i \cdot \text{SwiGLU}_i(x). \quad (21.3)$$

At training time different experts specialize on different parts of the data distribution resulting in a network that can model complicated data distributions. At test time, only experts relevant to the test sample are activated significantly reducing the computational load. In the context of the Mixtral model (and more generally LLM models), since the experts are applied independently on different input tokens, the Mixture of Experts module is used to replace fully connected layers of the architecture. We reproduce the following table of results

We observe that the Mixtral model is competitive with the LLaMA 2 70B model while using $5\times$ fewer active parameters during inference.

We now describe an improved version of the mixture of experts architecture implemented in the DeepSeekMoE model [393]. The crucial innovation is that the DeepSeekMoE uses finer-grained experts and isolates some experts as shared ones. Let u_t denote the FFN input of the t_{th} token. The authors compute the FFN output h'_t as follows:

$$h'_t = u_t + \sum_{i=1}^{N_s} \text{FFN}_i^{(s)}(u_t) + \sum_{i=1}^{N_r} g_{i,t} \text{FFN}_i^{(r)}(u_t) \quad (21.4)$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}} \quad (21.5)$$

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Top}_k(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r) \\ 0, & \text{otherwise} \end{cases} \quad (21.6)$$

$$s_{i,t} = \text{Sigmoid}(u_t^T e_i) \quad (21.7)$$

where N_s and N_r denote the numbers of shared experts and routed experts, respectively; $\text{FFN}_i^{(s)}(\cdot)$ and $\text{FFN}_i^{(r)}(\cdot)$ denote the i -th shared expert and the i -th routed expert, respectively; K_r denotes the number of activated routed experts; $g_{i,t}$ is the gating value for the i -th expert; $s_{i,t}$ is the token-to-expert affinity; e_i is the centroid vector of the i -th routed expert; and $\text{Top}_k(\cdot, K)$ denotes the set comprising K highest scores among the affinity scores calculated for the t -th token and all routed experts.

Importantly, routing diversity should be distinguished from Bayesian (posterior) or model uncertainty. Routing diversity arises from conditional composition — which experts are activated for a given input — and increases representational flexibility and conditional behavior. Posterior uncertainty, by contrast, refers to uncertainty over model parameters and predictions (epistemic uncertainty) and is typically estimated via Bayesian model averaging or independent deep ensembles. Router activations therefore do not by themselves provide calibrated epistemic uncertainty; when principled uncertainty estimates are required, routing-based models are best combined with ensemble or Bayesian techniques.

Another important innovation is the “Auxiliary-Loss-Free Load Balancing”. This is designed to avoid routing collapse [1508] while retaining model performance [511, 940, 1592]. Specifically the authors introduce a bias term b_i [1706] for each expert and add it to the corresponding affinity scores $s_{i,t}$ to determine the top-K routing:

$$g'_{i,t} = \begin{cases} s_{i,t} + b_i, & s_{i,t} + b_i \in \text{Top}_k(\{s_{j,t} + b_j | 1 \leq j \leq N_r\}, K_r) \\ 0, & \text{otherwise.} \end{cases} \quad (21.8)$$

Benchmark (Metric)		DeepSeek-V2	Qwen2.5	LLaMA-3.1	DeepSeek-V3
# Shots		72B	236B	72B	671B
# Activated Params		21B	72B	405B	37B
# Total Params		236B	72B	405B	671B
Pile-test (BPB)		0.606	0.638	0.542	0.548
BBH (EM)	3-shot	78.8	79.8	82.9	87.5
MMLU (EM)	5-shot	78.4	85.0	84.4	87.1
MMLU-Redux (EM)	5-shot	75.6	83.2	81.3	86.2
MMLU-Pro (EM)	5-shot	51.4	58.3	52.8	64.4
DROP (F1)	3-shot	80.4	80.6	86.0	89.0
ARC-Easy (EM)	25-shot	97.6	98.4	98.4	98.9
ARC-Challenge (EM)	25-shot	92.2	94.5	95.3	95.3
HellaSwag (EM)	10-shot	87.1	84.8	89.2	88.9
PIQA (EM)	0-shot	83.9	82.6	85.9	84.7
WinoGrande (EM)	5-shot	86.3	82.3	85.2	84.9
RACE-Middle (EM)	5-shot	73.1	68.1	74.2	67.1
RACE-High (EM)	5-shot	52.6	50.3	56.8	51.3
TriviaQA (EM)	5-shot	80.0	71.9	82.7	82.9
NaturalQuestions (EM)	5-shot	38.6	33.2	41.5	40.0
AGIEval (EM)	0-shot	57.5	75.8	60.6	79.6

Table 21.2: Comparison of English benchmark performance for DeepSeek-V2, Qwen2.5, LLaMA-3.1, and DeepSeek-V3, reproduced from [393].

Note that the bias term is only used for routing. The gating value, which will be multiplied with the FFN output, is still derived from the original affinity score $s_{i,t}$. During training, the bias is tuned dynamically.

The authors further propose a “Sequence-Wise Auxiliary Loss”. This loss is aimed at preventing extreme imbalance within any single sequence. Concretely the authors implement the following sequence-wise loss:

$$\ell_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i, \quad (21.9)$$

where the balance factor α is a hyper-parameter, which is assigned a small value, and where

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^T \mathbb{1}(\{s_{i,t} \in \text{Top}_k(\{s_{j,t} \mid 1 \leq j \leq N_r\}, K_r)\}) \quad (21.10)$$

and

$$P_i = \frac{1}{T} \sum_{t=1}^T s'_{i,t}, \quad \text{with} \quad s'_{i,t} = \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}}. \quad (21.11)$$

We observe in Table 21.2 that DeepSeek-V3 matches the performance of, or outperforms, the benchmark models while using significantly less activated parameters.

21.3 Applications to reinforcement learning

21.3.1 Reinforcement Learning from Human Feedback

The most known component of RL algorithms that is implemented with ensembles is Reinforcement Learning from Human Feedback (RLHF) [303]. RLHF is motivated by the difficulty in creating training signals for RL tasks such as open-ended text generation. To understand the difficulty we can compare with the case of image classification. For the task of image classification we can create a labeling of the dataset by first assigning an element of the set of natural numbers $\mathbb{N}$ to each class of the dataset, and train the network to predict this number for each labeled instance. For open-ended text it is much more difficult to assign an absolute number to the text quality as text quality has multiple degrees of freedom and includes subjective components. On the opposite, it is much easier to

	walker	hopper	swimmer	cheetah	ant	reacher	d-pendulum	pendulum
Ensemble	5000	5000	175	4500	−200	−3	7500	1250
No ensemble	3800	3900	60	3100	−300	−4	2500	1250

Table 21.3: Final reward of agents trained with RLHF on MuJoCo tasks before and after removing ensembling. We observe that ensembling results in significant improvements in final reward. Christiano et al. [303].

compare two texts and rank one as better than the other. RLHF utilizes such a ranking by human evaluators to learn a function (the reward function) that assigns consistent rankings to open-ended text. This function is then used to improve a policy given a state.

Let $\tau^A = (s, a^A)$ and $\tau^B = (s, a^B)$ be two (state, action) pairs and a reward model parametrized by ϕ . Human labelers provide pairwise preferences, such as:

$$\tau^A \succ \tau^B \quad \text{or} \quad \tau^B \succ \tau^A. \quad (21.12)$$

The learned reward function $\hat{r}_\phi$ produces a scalar reward for pair:

$$R(\tau) = \hat{r}_\phi(s, a). \quad (21.13)$$

The reward model assigns a probability that the question-answer pair τ^A is preferred over τ^B using a softmax:

$$P[\tau^A \succ \tau^B] = \frac{\exp(R(\tau^A))}{\exp(R(\tau^A)) + \exp(R(\tau^B))}. \quad (21.14)$$

Given a dataset of human preferences, the reward model is trained to minimize the following cross-entropy loss:

$$\ell_\phi = - \sum_{(\tau^A, \tau^B, \mu)} [\mu \log P[\tau^A \succ \tau^B] + (1 - \mu) \log P[\tau^B \succ \tau^A]], \quad (21.15)$$

where $\mu = 1$ (resp. $\mu = 0$) if the human preferred τ^A (resp. τ^B), and $\mu = 0.5$ if the preference was a tie. The learned reward model is then used in standard reinforcement learning (e.g., PPO) to train a policy π_θ :

$$\max_{\theta} \mathbb{E}_{q \sim \mathcal{D}, q \sim \pi_\theta(q)} [\hat{r}_\phi(q, a)] \quad (21.16)$$

where $\mathcal{D}$ is the query distribution.

Labelling huge amounts of data is a significant and expensive bottleneck for training RL models. It is thus crucial to choose labeling examples judiciously. For this reason instead of fitting a single reward model to different preference triplets, the original RLHF paper proposed fitting an ensemble of reward models. These can then be used to estimate a measure of uncertainty about triplets, and choose for labelling only the ones with high uncertainty. Concretely, the algorithm is as follows [303]:

- Fit an ensemble of predictors, each trained on $|D|$ triples sampled from D with replacement. The estimate r is defined by independently normalizing each of these predictors and then averaging the results.
- A fraction of $1/e$ of the data is held out to be used as a validation set for each predictor. Use ℓ_2 regularization and adjust the regularization coefficient to keep the validation loss between 1.1 and 1.5 times the training loss. In some domains we also apply dropout for regularization.
- Instead of applying a softmax directly, assume there is a 10% chance that the human responds uniformly at random. This adjustment is made because human raters have a constant probability of making an error, which doesn't decay to 0 as the difference in reward difference becomes extreme.
- Decide how to query preferences based on an approximation to the uncertainty in the reward function estimator: sample a large number of pairs of trajectory segments of length k , use each reward predictor in the ensemble to predict which segment will be preferred from each pair, and then select those trajectories for which the predictions have the highest variance across ensemble members.

We close by noting that RLHF is a crucial component of finetuning modern LLMs to make them more safe and reliable [76, 1273].

21.3.2 Model Ensemble in Model-Based RL

In the literature there exist various other applications of deep ensembles in RL [31, 216, 832, 952, 1193, 1825], these introduce ensembles both for model-based and model-free RL methods. We elaborate more on Kurutach et al. [952], Yu et al. [1825] which are model-based RL methods and use ensembles to capture the world model uncertainty.

We present simplified versions of the algorithms below. Yu et al. [1825] uses multiple Gaussians with learned means and covariances to model the relationship between future states and rewards. It implements the Model-based Offline Policy Optimization (MOPO) algorithm 23 for model-based RL. Similarly Kurutach et al. [952] uses multiple world models to generate synthetic samples for learning the policy. It implements the Model Ensemble Trust Region Policy Optimization algorithm (ME-TRPO) algorithm 24.

In more detail, MOPO [1825] operates in the offline RL regime. The approach begins by training an ensemble of dynamics models — each model, for example, learns a Gaussian distribution over next states (mean and covariance) given current state and action. The ensemble variance thus provides a quantitative measure of model uncertainty. When using the learned models to simulate transitions for policy learning, MOPO subtracts a penalty proportional to this uncertainty from the simulated rewards. This reward-penalised simulated MDP is shown to induce a lower bound on performance in the true environment, thereby guarding against model exploitation and distributional shift. Empirically, MOPO surpasses prior offline RL methods on standard benchmarks. Meanwhile, ME-TRPO [952] targets the standard online model-based RL setup, and likewise employs an ensemble of learned world models to account for areas of high model error. ME-TRPO generates synthetic rollouts from the ensemble, updates the policy via a trust-region method (ensuring stable updates), and uses the ensemble disagreement to filter or penalise untrustworthy model trajectories. By doing so, ME-TRPO reduces real-environment interactions while maintaining policy robustness. Both methods highlight how leveraging ensembles of world-models — and using their predictive variance or disagreement — enables uncertainty-aware policy optimisation in model-based RL.

Algorithm 23: MOPO instantiation with regularized probabilistic dynamics and ensemble uncertainty

Input: Reward penalty coefficient λ , rollout horizon h , rollout batch size b

Output: Trained policy π

// Train ensemble dynamics model

Train on batch data $\mathcal{D}_{\text{env}}$ an ensemble of N probabilistic dynamics

$\{T_i^b(s', r \mid s, a) = \mathcal{N}(\mu_i(s, a), \Sigma_i(s, a))\}_{i=1}^N$

Initialize policy π and empty replay buffer $\mathcal{D}_{\text{model}} \leftarrow \emptyset$

for $epoch = 1, 2, \dots$ **do**

 // Outer iteration of MBPO

for $= 1, 2, \dots, b$ **do**

 Sample state s_1 from $\mathcal{D}_{\text{env}}$ for rollout initialization

for $j = 1, 2, \dots, h$ **do**

 Sample action $a_j \sim \pi(s_j)$

 Randomly pick dynamics T^b from $\{T_i^b\}_{i=1}^N$ and sample $(s_{j+1}, r_j) \sim T^b(s_j, a_j)$

 Compute $\tilde{r}_j = r_j - \lambda \max_{i=1, \dots, N} \|\Sigma_i(s_j, a_j)\|_F$

 Add $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to $\mathcal{D}_{\text{model}}$

 Draw samples from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$ and update π

While the way in which ensembles are constructed differs significantly, we see that in both cases ensembles of models are used to maintain the world-model uncertainty and regularize the learning process. This in turn significantly improves the sample complexity and final rewards. We also reproduce here the reward results with and without ensembles for MOPO and MBPO for various Mujoco [1619] environments. We observe that ensembles improve both MOPO as well as the MBPO baseline.

21.4 Applications to weather forecasting

Another critical application of deep ensembles is in weather models. Ensemble methods provide important uncertainty estimates for weather models which can help in mitigating the effects of extreme weather conditions. Example works include Aizzadenesheli et al. [67], Chen et al. [269], Pathak et al. [1302], Weyn et al. [1744]. It has

Algorithm 24: Model Ensemble Trust Region Policy Optimization (ME-TRPO)

Input: Number of world models K
Output: Optimized policy π_θ
Initialize policy π_θ and world models $\{\hat{f}_{\phi_i}\}_{i=1}^K$; set dataset $\mathcal{D} \leftarrow \emptyset$
repeat
 Collect samples from real system f using π_θ and add them to $\mathcal{D}$
 Train all models on $\mathcal{D}$
 repeat
 Collect fictitious samples from $\{\hat{f}_{\phi_i}\}_{i=1}^K$ using π_θ
 Update π_θ with TRPO on fictitious samples
 Estimate performances $\hat{\eta}(\theta; \phi_i)$ for $i = 1, \dots, K$
 until performances stop improving
until π_θ performs well in real environment f

Method	halfcheetah-mixed	walker2d-mixed	halfcheetah-jump	ant-angle
MOPO w/ ens.	6448.7 $\pm$ 115	1923.6 $\pm$ 752	3577.3 $\pm$ 461	2256.0 $\pm$ 288
MOPO w/o ens.	6405.8 $\pm$ 35	1916.4 $\pm$ 611	4016.6 $\pm$ 144	2530.9 $\pm$ 137
MBPO w/ ens.	5598.4 $\pm$ 1285	1021.8 $\pm$ 586	2971.4 $\pm$ 1262	13.6 $\pm$ 65
MBPO w/o ens.	2247.2 $\pm$ 581	500.3 $\pm$ 34	-68.7 $\pm$ 1936	-720.1 $\pm$ 728

Table 21.4: Performance comparison of different methods on various tasks MOPO [1825] and MBPO [832] reproduced from the original papers. We observe that ensembles improve both MOPO as well as the MBPO baseline.

been long established that ensembles of traditional numerical methods, using different initial conditions, result in improved weather estimates. The key innovation of most deep learning papers is in the speed of computation for each ensemble member. Deep neural networks can significantly speed up the solution of the Partial Differential Equations needed to predict weather patterns. This in turn allows to use significantly larger ensembles than what would have otherwise been possible. For example the speed of FourCastNet [1302] enables the creation of rapid and inexpensive large-ensemble forecasts with thousands of ensemble-members for improving probabilistic forecasting, and is orders of magnitude faster than traditional approaches.

21.5 Applications to medical data

There is a large number of works that utilize ensembles for medical applications [32, 775, 841, 1718, 1746]. These range from using ensembles for reliable predictions in clinical time series [1746] to reliable detection of Alzheimer’s disease [841]. We elaborate more on Wickstrøm et al. [1746]. Since this work operates on time-series data, it serves to highlight the diverse settings in which deep ensembles improve test metrics. Here the authors explore the class activation mapping (CAM) method [1870], for explainable deep learning prediction. CAM was originally developed for image data, but has also been shown to be applicable for temporal data such as clinical time series [58, 1717]. By utilizing DNNs with a particular processing structure, CAM can assign a score to each time step in an input that indicates how important that time step is for a given prediction. Example cases of time-series data are mortality prediction, detection of infections, and patient treatment trajectory prediction.

Let $w_{c,k}$ denote the weight connecting the k th filter in the last convolutional block with the neuron corresponding to class c in the output layer, and $z_{k,t}$ denote the activation at time step t produced by the k th filter in the last convolutional block of the FCN. Then, the input to the final softmax function in the output layer can be expressed as:

$$g_c = \sum_{k=1}^K w_{c,k} \sum_{t=1}^T z_{k,t} = \sum_{t=1}^T r_{c,t}, \quad (21.17)$$

where K is the number of filters in the last convolutional block, T is the length of the time series, and $r_{c,t}$ denotes the relevance score of time step t for class c :

$$r_{c,t} = \sum_{k=1}^K w_{c,k} z_{k,t}. \quad (21.18)$$

Removing all negative relevance scores can be achieved by modifying Equation (21.18) as follows:

$$r_{c,t} = \max \left(0, \sum_{k=1}^K w_{c,k} z_{k,t} \right). \quad (21.19)$$

To calculate the ensemble mean and standard deviation of the relevance scores, let each CNN be parametrized by a set of parameters $\{\theta_1, \dots, \theta_M\}$, where M is the number of models in the ensemble. Then, when considering the positive relevance scores from Equation (21.19), the mean relevance across the ensemble is defined as:

$$\mu_r = \frac{1}{M} \sum_{m=1}^M r_{\theta_m}, \quad (21.20)$$

and the standard deviation across the ensemble is defined as:

$$\sigma_r = \sqrt{\frac{1}{M-1} \sum_{m=1}^M (r_{\theta_m} - \mu_r)^2}. \quad (21.21)$$

In all experiments the authors scale the relevance scores between 0 and 1 using the following equation:

$$r_c \leftarrow \frac{r_c - \min(r_c)}{\max(r_c) - \min(r_c)}. \quad (21.22)$$

If the standard deviation is above some threshold, the relevance scores are set to zero. This can be formulated as:

$$\tilde{\mu}_{r_t} = \begin{cases} \mu_{r_t}, & \text{if } \sigma_{r_t} < \varepsilon, \\ 0, & \text{else,} \end{cases} \quad (21.23)$$

where the threshold ε is chosen to fit the particular data.

This approach results in significant improvement in explainability over the single network. To calculate how accurate a model is at locating relevant time steps, the authors compare the k most relevant time steps for a prediction with the known most relevant time steps. For a given sample i , let

$$Y_i = \{y_1^{(i)}, \dots, y_{N_r}^{(i)}\}, \quad (21.24)$$

denote the set of relevant points, where N_r is the number of known relevant time steps in the time series, and let $R_i(k)$ denote the k most relevant time steps for the prediction of the model.

Perfect relevance accuracy is achieved when all elements of Y_i are contained in $R_i(k)$. For a given sample i , relevance accuracy can be calculated as:

$$\text{RelevanceAccuracy}_i(k) = \frac{|R_i(k) \cap Y_i|}{|Y_i|}. \quad (21.25)$$

For a given sample i , computing the relevance consistency across M models can then be achieved by:

$$\text{RelevanceConsistency}_i(k) = \frac{1}{M} \sum_{m,n=1}^M \frac{|R_i^{(m)}(k) \cap R_i^{(n)}(k)|}{k}. \quad (21.26)$$

We reproduce here some results on the EGG200 dataset [374] which consists of EGG time-series that trace the electrical activity recorded during a single heartbeat. The task is to discriminate between normal heartbeats and those exhibiting signs of myocardial infarction (heart attack). Table 21.5 shows the relevance consistency of 10 single and 10 ensemble models on both datasets.

We observe that ensembles significantly improve the relevance consistency of the relevance scores.

21.6 Applications to autonomous driving

Ensemble learning has become an important strategy for improving the reliability, robustness, and uncertainty estimation of autonomous driving systems, particularly for trajectory forecasting in safety-critical environments. By combining the predictions of multiple models, ensembles reduce overconfidence, improve generalization, and

Top k	Single	Ensemble
$k = 5$	0.57 ± 0.15	0.82 ± 0.07
$k = 7$	0.60 ± 0.15	0.84 ± 0.07
$k = 10$	0.63 ± 0.14	0.85 ± 0.06
$k = 15$	0.67 ± 0.13	0.88 ± 0.05

Table 21.5: Performance comparison of Single vs. Ensemble models on ECG200 dataset. The numbers indicate the statistical significance at a significance level of 0.01.

provide more accurate representations of the multimodal nature of future vehicle and pedestrian behaviors. Traditional deep ensembles, however, incur substantial computational and memory costs, limiting their deployment in real-time autonomous driving applications. Recent research [957, 958] addresses this challenge through efficient ensemble architectures, such as lightweight transformer ensembles and packed or grouped subnetworks, which retain the predictive diversity and uncertainty calibration of conventional ensembles while significantly reducing computational overhead. These approaches enable autonomous vehicles to generate more reliable long-term trajectory predictions, better quantify uncertainty, and make safer planning and decision-making under complex and ambiguous traffic scenarios.

21.7 Patents on deep ensembles

Patents on deep ensembles demonstrate the growing relevance of ensemble-based deep learning techniques across diverse real-world applications. For example, [685] illustrates how ensemble-based approaches can improve decision-making in autonomous vehicles by incorporating uncertainty estimates alongside object detection outputs, enabling safer and more reliable predictions. Similarly, [222] demonstrates the use of ensemble deep learning models for enhancing digital experiences by combining multiple machine learning estimators with neural networks to generate improved recommendations. Furthermore, [1484] highlights how ensemble machine learning techniques can improve future event detection by combining models with different prediction characteristics to balance accuracy and timeliness. These patents collectively demonstrate the practical relevance of deep ensembles in improving robustness, uncertainty estimation, adaptability, and predictive performance across complex real-world applications.

Chapter 22

PAC-Bayesian deep neural network ensembles

PAC-Bayesian analysis is a frequentist technique for deriving generalisation bounds for randomised classifiers. As a special case, PAC-Bayesian analysis can be used to provide generalisation bounds for Bayesian classifiers. We provide a basic introduction to PAC-Bayesian analysis and discuss similarities and distinctions from Bayesian learning. We then discuss ensemble classifiers and weighted majority votes and provide first and second order oracle bounds on the expected loss of weighted majority votes. The oracle bounds relate the expected loss of a weighted ensemble to the expected loss of a randomised classifier. Since the expected loss of a randomised classifier is an unknown (oracle) quantity, the bounds are called oracle bounds. We apply PAC-Bayesian analysis to bound the expected loss of randomised classifiers in terms of their empirical loss, thereby converting oracle bounds into empirical ones. The empirical bounds can be used to provide generalisation guarantees, as well as optimise weighting of weighted majority votes.

22.1 A basic introduction to PAC-Bayesian analysis

PAC-Bayesian analysis was invented by Shawe-Taylor and Williamson [1507] and McAllester [1154]. Their motivation was to introduce a possibility to incorporate prior knowledge within the frequentist learning framework. The name *PAC-Bayesian* is derived from *PAC*, standing for *Probably Approximately Correct* learning framework proposed by Valiant [1652], and *Bayesian*, which refers to the method's ability to incorporate prior knowledge in the analysis. PAC-Bayesian analysis is a technique for deriving frequentist generalisation bounds for randomised classifiers. A *randomised classifier* defined by a distribution ρ on a set of parameters Θ acts by sampling a new parameter vector $\theta \in \Theta$ according to $\rho(\theta)$ each time it classifies a new data point.

Definition 22.1 (Randomised Classifier). *Let ρ be a distribution over Θ . A randomised classifier associated with ρ (and named ρ) is defined by the following protocol. At each prediction round it:*

1. *Samples $\theta \in \Theta$ according to $\rho(\theta)$*
2. *Observes x*
3. *Returns $f(x, \theta)$*

We use $\ell(f(x, \theta), y)$ to denote a loss function for predicting $f(x, \theta)$ when the true label is y . In contrast to most other chapters in the book, where ℓ is the negative log-likelihood, most results in this chapter assume that $\ell(\cdot, \cdot)$ is bounded in the $[0, 1]$ interval. For example, $\ell(\cdot, \cdot)$ could be the zero-one loss function. We use $\mathcal{L}(\theta) = \mathbb{E}[\ell(f(x, \theta), y)]$ to denote the expected loss of θ and $\hat{\mathcal{L}}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \ell(f(x_i, \theta), y_i)$ the empirical loss of θ . Then the expected loss of ρ is $\mathbb{E}_{\theta \sim \rho}[\mathcal{L}(\theta)]$ and the empirical loss of ρ is $\mathbb{E}_{\theta \sim \rho}[\hat{\mathcal{L}}(\theta, \mathcal{D})]$. Whenever it causes no confusion, we will shorten the notation to $\mathbb{E}_{\rho}[\mathcal{L}(\theta)]$ and $\mathbb{E}_{\rho}[\hat{\mathcal{L}}(\theta, \mathcal{D})]$.

There exist numerous forms of PAC-Bayesian bounds. We start with one due to McAllester [1155], which is convenient for interpreting the nature of PAC-Bayes. It is a relaxation of a tighter bound of Seeger [1483] that will be shown later in Theorem 22.3, together with a proof of both results.

Theorem 22.1 (Interpretation-friendly PAC-Bayesian inequality). *Assume that the loss function ℓ is bounded in the $[0, 1]$ interval. For any “prior” distribution π over Θ that is independent of $\mathcal{D}$ and any $\delta \in (0, 1]$*

$$\mathbb{P} \left(\forall \rho : \mathbb{E}_\rho[\mathcal{L}(\theta)] \leq \mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})] + \sqrt{\frac{2\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})] \left(\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta} \right)}{N}} + \frac{2 \left(\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta} \right)}{N} \right) \geq 1 - \delta. \quad (22.1)$$

We emphasise that Theorem 22.1 is a frequentist generalisation bound and the random variable inside the probability statement is $\mathcal{D}$ (and thus $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$), but not θ (and thus not $\mathbb{E}_\rho[\mathcal{L}(\theta)]$).

We can restate the result of the theorem verbally by saying that with probability at least $1 - \delta$, for all “posterior” distributions ρ

$$\mathbb{E}_\rho[\mathcal{L}(\theta)] \leq \mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})] + \sqrt{\frac{2\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})] \left(\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta} \right)}{N}} + \frac{2 \left(\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta} \right)}{N}. \quad (22.2)$$

We wrote “prior” and “posterior” in quotation marks, because the semantic meaning of PAC-Bayesian priors and posteriors differs from Bayesian priors and posteriors. In PAC-Bayes a prior distribution can be any distribution that does not depend on the data $\mathcal{D}$ used in the computation of the bound. It can be used to represent a prior belief on the performance $\mathcal{L}(\theta)$ of different parameters θ , but the bound holds irrespective of the source of π . And the bound holds for all posterior distributions ρ . In particular, it allows for the selection of ρ based on the data $\mathcal{D}$. The bound also holds for the Bayes posterior, but is not restricted to it.

Since Theorem 22.1 holds for all posterior distributions ρ , it can be used to find a distribution ρ that yields the best guarantee on the expected loss $\mathbb{E}_\rho[\mathcal{L}(\theta)]$ according to the bound in equation (22.2). A distribution ρ that minimises the bound should balance between placing a high mass on the parameters θ with a low empirical risk $\hat{\mathcal{L}}(\theta, \mathcal{D})$ in order to minimise the $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$ term, and staying close to the prior distribution π in the D_{KL} sense to keep $\text{D}_{\text{KL}}(\rho \parallel \pi)$ under control. The $\text{D}_{\text{KL}}(\rho \parallel \pi)$ term measures the complexity of a distribution ρ . If $\rho = \pi$, then $\text{D}_{\text{KL}}(\rho \parallel \pi) = 0$, meaning that if the posterior takes no information from the sample and remains the same as the prior, then the complexity is zero. While the bound holds for the Bayes posterior, it is typically *not* the Bayes posterior that minimises the right hand side of equation (22.2).

The prior distribution π is a tool to steer learning before revealing the data. If the prior allocates high probability mass to a subset of the parameter space that happens to work well on the data (happens to achieve low $\hat{\mathcal{L}}(\theta, \mathcal{D})$ after the data is revealed), the posterior ρ minimising the right hand side of equation (22.2) will not need to deviate much from π , leading to a small $\text{D}_{\text{KL}}(\rho \parallel \pi)$ term and a tight bound. Otherwise the bound might be loose, but it always remains a valid bound.

The bound in Theorem 22.1 provides a “fast convergence rate”. Specifically, if for some ρ the empirical risk $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$ happens to be of order $O(\frac{1}{N})$, the last two terms in equation (22.2) scale with $\frac{1}{N}$, which is known as “fast rate”, as opposed to scaling of the middle term with $\frac{1}{\sqrt{N}}$ for large $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$, which is known as “slow rate”. In other words, when $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$ is small, with high probability it is even closer to $\mathbb{E}_\rho[\mathcal{L}(\theta)]$ than when it is large. In the context of deep learning, where reaching zero empirical risk is often possible, it provides an extra benefit for allocating higher posterior mass to well-performing parameters and bringing $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$ close to zero.

Theorem 22.1 is convenient for explaining the nature of PAC-Bayes, but it is not the most convenient for optimisation, because the bound is not convex in ρ . The following result by Thiemann et al. [1599] provides an optimisation-friendly version of the bound. Similar to Theorem 22.1, it is a relaxation of Theorem 22.3 that we discuss later, and it is at least as tight as Theorem 22.1 (see Tolstikhin and Seldin [1620]).

Theorem 22.2 (Optimisation-friendly PAC-Bayesian inequality). *Assume that the loss function ℓ is bounded in the $[0, 1]$ interval. For any prior distribution π over Θ that is independent of $\mathcal{D}$ and any $\delta \in (0, 1]$*

$$\mathbb{P} \left(\forall \rho : \mathbb{E}_\rho[\mathcal{L}(\theta)] \leq \min_{\lambda \in (0, 2)} \left(\frac{\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]}{1 - \frac{\lambda}{2}} + \frac{\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta}}{\lambda(1 - \frac{\lambda}{2})N} \right) \right) \geq 1 - \delta. \quad (22.3)$$

In other words, with probability at least $1 - \delta$ over the draw of a sample $\mathcal{D}$, for all distributions ρ over Θ and all $\lambda \in (0, 2)$

$$\mathbb{E}_\rho[\mathcal{L}(\theta)] \leq \frac{\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]}{1 - \frac{\lambda}{2}} + \frac{\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta}}{\lambda(1 - \frac{\lambda}{2})N}. \quad (22.4)$$

For a fixed λ the right hand side of the inequality (22.4) is convex in ρ (because $\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$ is linear in ρ and $\text{D}_{\text{KL}}(\rho \parallel \pi)$ is convex in ρ), and for a fixed ρ it is convex in λ (see Tolstikhin and Seldin [1620]). Therefore, the bound in Theorem 22.2 is amenable to alternating minimisation. For a fixed λ the bound is minimised by

$$\rho_\lambda^*(\theta) = \frac{\pi(\theta)e^{-\lambda N \hat{\mathcal{L}}(\theta, \mathcal{D})}}{\mathbb{E}_\pi[e^{-\lambda N \hat{\mathcal{L}}(\theta', \mathcal{D})}]} \quad (22.5)$$

(we leave the derivation as an exercise) and for a fixed ρ the bound is minimised by

$$\lambda_\rho^* = \frac{2}{\sqrt{\frac{2N\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]}{\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta}}} + 1} + 1 \quad (22.6)$$

(see Tolstikhin and Seldin [1620]). Alternating application of the update rules (22.5) and (22.6) leads to a local minimum of the bound (22.4). We note that the bound (22.4) is not jointly convex in ρ and λ , which would have been a sufficient, but not a necessary condition for convergence to the global minimum. Thiemann et al. [1599] provide sufficient conditions under which the bound (22.4) is strongly quasi-convex in ρ and λ , which guarantees convergence of alternating minimisation to the global minimum, as well as examples of cases where the strong quasi-convexity conditions are violated and the procedure only converges to a local minimum.

22.1.1 Similarities and differences between PAC-Bayesian analysis and Bayesian learning

Theorems 22.1 and 22.2 provide sufficient background to compare PAC-Bayesian analysis to Bayesian learning. We start with the similarities and then discuss the differences.

Similarities to Bayesian learning

1. Similar to Bayesian learning, PAC-Bayesian analysis provides an explicit way to incorporate prior knowledge via a prior distribution π .
2. Similar to Bayesian learning, it also allows sequential updates of the prior knowledge π as more data become available. We refer to the work of Wu et al. [1781] on Recursive PAC-Bayes, which is a procedure for sequential prior knowledge update without loss of confidence information.

Differences with Bayesian learning

1. PAC-Bayesian analysis is a frequentist learning framework. The random variables in PAC-Bayesian analysis are the observations (the data $\mathcal{D}$), but not the parameters θ . The prior π in PAC-Bayesian analysis is just an auxiliary construction for deriving the bound (as will be formally shown in the next section). It is not a prior belief in the Bayesian sense on the distribution from which the parameters of a “true” model generating the data were sampled. PAC-Bayesian bounds are valid for any choice of the prior. The choice of a prior impacts the tightness of the bound, but never compromises its validity, as long as the prior is independent of the data on which the bound is computed.
2. PAC-Bayesian posteriors are averaging distributions $\rho(\theta)$ and the bounds hold for all possible posterior distributions $\rho(\theta)$. Bayesian posteriors are conditional distributions $\rho(\theta|\mathcal{D})$. Given a Bayesian prior $\pi(\theta)$ and a dataset $\mathcal{D}$, the Bayesian posterior is uniquely determined. Thus, the semantics of the PAC-Bayesian and Bayesian posteriors are different.
3. The loss function ℓ is a central element in PAC-Bayesian analysis, because the PAC-Bayesian objective is to bound the expected risk $\mathbb{E}_\rho[\mathcal{L}(\theta)]$. To work with a PAC-Bayesian bound, one has to define a loss function, and a change of the loss function changes the minimiser ρ of the bound. In contrast, the definition of the Bayes rule, $\rho(\theta|\mathcal{D}) = \pi(\theta)\mathbb{P}(\mathcal{D}|\theta)/\mathbb{P}(\mathcal{D})$, is not based on a loss function.

4. To draw further parallels and distinctions between Bayesian and PAC-Bayesian learning, assume that $f(x, \theta)$ returns a probability distribution on $\mathcal{Y}$ and that the loss function ℓ is the negative log-likelihood. Then the conditional data probability $\mathbb{P}(\mathcal{D}|\theta)$ can be written as $\mathbb{P}(\mathcal{D}|\theta) = e^{-N\hat{\mathcal{L}}(\theta, \mathcal{D})}$ and the Bayesian posterior can be written as $\rho(\theta) = \frac{\pi(\theta)e^{-N\hat{\mathcal{L}}(\theta, \mathcal{D})}}{\mathbb{E}_{\pi}\left[e^{-N\hat{\mathcal{L}}(\theta', \mathcal{D})}\right]}$, which corresponds to taking $\lambda = 1$ in equation (22.5). This connection is useful to underline two important differences.
 - (a) PAC-Bayesian analysis relies on convergence of empirical estimates of the risk $\hat{\mathcal{L}}(\theta, \mathcal{D})$ to their expectations $\mathcal{L}(\theta)$. The convergence holds for bounded loss functions and, therefore, the assumption that the loss function is bounded in the $[0, 1]$ interval is common in the PAC-Bayesian literature. Unbounded loss functions, including the negative log-likelihood, do not guarantee such convergence by default and require extra care. For example, smoothing of the posterior distribution, as done by Seldin and Tishby [1486], or introduction of assumptions on the data generating distribution that would guarantee convergence of empirical means to their expectations. Therefore, in case of unbounded loss functions application of PAC-Bayesian bounds to provide generalisation guarantees for Bayesian posteriors may demand extra analysis, assumptions on the data generating distribution, and/or posterior adjustments. Since Bayesian inference is not directly concerned with concentration of empirical estimates of the losses (and loss functions in general), unboundedness of the loss function is not a direct concern for it.
 - (b) The second distinction is that the optimal value of λ in PAC-Bayesian bounds is typically smaller than 1, meaning that they weigh prior belief $\pi(\theta)$ and evidence differently from Bayesian learning, and if the loss is not the negative log-likelihood, then the shape of the optimal posterior might be altogether completely different.
5. The Bayes rule requires access to the prior $\pi(\theta)$ to compute the posterior $\rho(\theta|\mathcal{D})$. In PAC-Bayesian bounds π appears only in $D_{\text{KL}}(\rho||\pi)$ and, therefore, it is sufficient to have an upper bound on $D_{\text{KL}}(\rho||\pi)$ to work with the bounds. For example, Lever et al. [1015, 1016] define $\pi(\theta)$ in terms of the true data generating distribution. Such $\pi(\theta)$ is valid, because it does not depend on the data, but it is not accessible to the learner. Nevertheless, Lever et al. [1015, 1016] show that if both the prior and the posterior are Gibbs distributions, it is possible to bound $D_{\text{KL}}(\rho||\pi)$ using a self-bounding technique and work with the bound without direct access to $\pi(\theta)$.

22.1.2 PAC-Bayes-kl inequality

Now we present the PAC-Bayes-kl inequality due to Seeger [1483] (with a slight improvement by Maurer [1151]). The inequality is at the basis of Theorems 22.1 and 22.2 we have discussed earlier, and will be followed by a proof. In order to proceed, we need to define the binary kl-divergence, which is a shorthand for the standard KL divergence (see Chapter 12.1) between two Bernoulli distributions.

Definition 22.2 (Binary kl-divergence). *Let p and q be biases of two Bernoulli random variables. The binary D_{kl} divergence is defined as:*

$$D_{\text{kl}}(p||q) = D_{\text{KL}}([1-p, p]||[1-q, q]) = p \ln \frac{p}{q} + (1-p) \ln \frac{1-p}{1-q}. \quad (22.7)$$

Pay attention that the parameters of $D_{\text{KL}}(\cdot||\cdot)$ are two probability distributions, whereas the parameters of $D_{\text{kl}}(\cdot||\cdot)$ are two numbers in the $[0, 1]$ interval. PAC-Bayes-kl inequality presented next involves binary kl divergence between the empirical and the expected loss of a randomised classifier ρ (which are two numbers in $[0, 1]$) and the standard KL divergence between the posterior and the prior (which are two distributions).

Theorem 22.3 (PAC-Bayes-kl inequality). *Assume that the loss function ℓ is bounded in the $[0, 1]$ interval. For any prior distribution π over Θ that is independent of $\mathcal{D}$ and any $\delta \in (0, 1]$:*

$$\mathbb{P}\left(\exists \rho : D_{\text{kl}}\left(\mathbb{E}_{\rho}\left[\hat{\mathcal{L}}(\theta, \mathcal{D})\right] \middle\| \mathbb{E}_{\rho}[\mathcal{L}(\theta)]\right) \geq \frac{D_{\text{KL}}(\rho||\pi) + \ln \frac{2\sqrt{N}}{\delta}}{N}\right) \leq \delta. \quad (22.8)$$

Theorem 22.1 follows from Theorem 22.3 by the refined Pinsker's inequality [1140, 1141, 1455], [196, Lemma 8.4]. We leave the proof of this statement as an exercise.

Lemma 22.1 (Refined Pinsker's inequality). *If $D_{\text{kl}}(p||q) \leq \varepsilon$ then*

$$q \leq p + \sqrt{2p\varepsilon} + 2\varepsilon. \quad (22.9)$$

Theorem 22.2 also follows from Theorem 22.3 by another form of the refined Pinsker's inequality.

Lemma 22.2 (Variational form of the refined Pinsker's inequality). *If $D_{\text{kl}}(p||q) \leq \varepsilon$ then*

$$q \leq \min_{\lambda \in (0,2)} \left(\frac{p}{1 - \frac{\lambda}{2}} + \frac{\varepsilon}{\lambda(1 - \frac{\lambda}{2})} \right). \quad (22.10)$$

The proof of PAC-Bayes-kl inequality is based on change of measure inequality, which acts as a replacement of the union bound in continuous spaces. The change of measure inequality uses Jensen's inequality, which we present first.

Theorem 22.4 (Jensen's inequality). *If g is a convex function and Z a random variable, then*

$$\mathbb{E}[g(Z)] \geq g(\mathbb{E}[Z]). \quad (22.11)$$

We refer the reader to Cover and Thomas [337] for a proof. Next, we present the change of measure inequality.

Theorem 22.5 (Change of measure inequality). *For any measurable function $g(\theta)$ on Θ and any distributions ρ and π :*

$$\mathbb{E}_{\theta \sim \rho}[g(\theta)] \leq D_{\text{KL}}(\rho||\pi) + \ln \mathbb{E}_{\theta \sim \pi}[e^{g(\theta)}]. \quad (22.12)$$

Proof.

$$\mathbb{E}_{\theta \sim \rho}[g(\theta)] = \mathbb{E}_{\theta \sim \rho} \left[\ln \left(\frac{\rho(\theta)}{\pi(\theta)} \times e^{g(\theta)} \times \frac{\pi(\theta)}{\rho(\theta)} \right) \right] \quad (22.13)$$

$$= D_{\text{KL}}(\rho||\pi) + \mathbb{E}_{\theta \sim \rho} \left[\ln \left(e^{g(\theta)} \times \frac{\pi(\theta)}{\rho(\theta)} \right) \right] \quad (22.14)$$

$$\leq D_{\text{KL}}(\rho||\pi) + \ln \mathbb{E}_{\theta \sim \rho} \left[e^{g(\theta)} \times \frac{\pi(\theta)}{\rho(\theta)} \right] \quad (22.15)$$

$$= D_{\text{KL}}(\rho||\pi) + \ln \mathbb{E}_{\theta \sim \pi} [e^{g(\theta)}], \quad (22.16)$$

where the inequality in the third line is by Jensen's inequality. Note that there is nothing probabilistic in the statement of the theorem – it is a deterministic result. $\square$

The change of measure inequality can also be obtained from the variational formula by Donsker and Varadhan [441], which states that

$$\ln \mathbb{E}_{\theta \sim \pi} [e^{g(\theta)}] = \sup_{\rho \ll \pi} (\mathbb{E}_{\theta \sim \rho} [g(\theta)] - D_{\text{KL}}(\rho||\pi)), \quad (22.17)$$

where $\rho \ll \pi$ is used to denote absolute continuity of ρ with respect to π , and $\ln \mathbb{E}_{\theta \sim \pi} [e^{g(\theta)}]$ is assumed to be finite. Note that taking $g(\theta) = \ln p(\mathcal{D}|\theta)$ in the change of measure inequality recovers the ELBO (Evidence Lower Bound) discussed in the earlier chapters.

The change of measure inequality is used in the PAC-Bayes lemma, which is the foundation of most PAC-Bayesian bounds. The PAC-Bayes lemma also uses Markov's inequality, so we start with it.

Theorem 22.6 (Markov's inequality). *For any non-negative random variable Z and $\varepsilon > 0$*

$$\mathbb{P}(Z \geq \varepsilon) \leq \frac{\mathbb{E}[Z]}{\varepsilon}. \quad (22.18)$$

Equivalently (taking $\delta = \frac{\mathbb{E}[Z]}{\varepsilon}$), for any $\delta \in (0, 1]$:

$$\mathbb{P}\left(Z \geq \frac{\mathbb{E}[Z]}{\delta}\right) \leq \delta. \quad (22.19)$$

We refer the reader to Seldin [1485] for a proof and move on to the PAC-Bayes lemma.

Lemma 22.3 (PAC-Bayes lemma). *For any measurable function $g : \Theta \times (\mathcal{X} \times \mathcal{Y})^N \rightarrow \mathbb{R}$ and any distribution π on Θ that is independent of the sample $\mathcal{D}$*

$$\mathbb{P}\left(\exists \rho : \mathbb{E}_{\theta \sim \rho}[g(\theta, \mathcal{D})] \geq D_{\text{KL}}(\rho||\pi) + \ln \frac{\mathbb{E}_{\theta \sim \pi}[\mathbb{E}_{\mathcal{D}}[e^{g(\theta, \mathcal{D})}]]}{\delta}\right) \leq \delta, \quad (22.20)$$

where the probability is with respect to the draw of the sample $\mathcal{D}$ and $\mathbb{E}_{\mathcal{D}}[\cdot]$ is the expectation with respect to the draw of $\mathcal{D}$.

Proof.

$$\mathbb{P}\left(\exists \rho : \mathbb{E}_\rho[g(\theta, \mathcal{D})] \geq D_{\text{KL}}(\rho \| \pi) + \ln \frac{\mathbb{E}_\pi[\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}]]}{\delta}\right) \quad (22.21)$$

$$\leq \mathbb{P}\left(\mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}] \geq \frac{\mathbb{E}_\pi[\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}]]}{\delta}\right) \quad (22.22)$$

$$= \mathbb{P}\left(\mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}] \geq \frac{\mathbb{E}_\mathcal{D}[\mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}]]}{\delta}\right) \quad (22.23)$$

$$\leq \delta, \quad (22.24)$$

where equation (22.22) follows by the change of measure inequality (for a fixed π , we have $\mathbb{E}_\rho[g(h, S)] - D_{\text{KL}}(\rho \| \pi) \leq \ln \mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}]$ for all ρ); in equation (22.23) we can exchange the order of expectations, because π is independent of $\mathcal{D}$; and in the last step we apply Markov's inequality to the random variable $Z = \mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}]$.

An important point is that the change-of-measure inequality bounds $\mathbb{E}_\rho[g(h, S)] - D_{\text{KL}}(\rho \| \pi)$ for all posterior distributions ρ by a single quantity $\ln \mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}]$ in a deterministic way, and the probabilistic argument is applied to a single random variable $\mathbb{E}_\pi[e^{g(\theta, \mathcal{D})}]$, avoiding the need to take a union bound. In this way, the change of measure inequality acts as a replacement of the union bound. $\square$

Various PAC-Bayesian inequalities are obtained from the PAC-Bayes lemma by making suitable choices of the function $g(\theta, \mathcal{D})$. A key consideration in the choice of $g(\theta, \mathcal{D})$ is the possibility to bound $\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}]$. For example, Seldin et al. [1487] have used Hoeffding's lemma and Bernstein's lemma to derive PAC-Bayes-Hoeffding and PAC-Bayes-Bernstein inequalities, respectively. In PAC-Bayes-kl the choice is $g(\theta, \mathcal{D}) = N D_{\text{kl}}(\hat{\mathcal{L}}(\theta, \mathcal{D}) \| \mathcal{L}(\theta))$. The following lemma is used to control the expectation of the exponent of g .

Lemma 22.4 (kl Lemma). *Let $Z_1, \dots, Z_N$ be independent identically distributed random variables with $Z_1 \in [0, 1]$, $\mathbb{E}[Z_1] = p$, and $\hat{p} = \frac{1}{N} \sum_{i=1}^N Z_i$. Then*

$$\mathbb{E}[e^{N D_{\text{kl}}(\hat{p} \| p)}] \leq 2\sqrt{N}. \quad (22.25)$$

We refer the reader to Maurer [1151], Seldin [1485] for a proof and to Seldin [1485] for examples of other choices of g .

Taking $Z_i = \ell(f(x_i, \theta), y_i)$, assuming that the loss function ℓ is bounded in $[0, 1]$, and applying Lemma 22.4, to the above choice of g , we obtain $\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}] \leq 2\sqrt{N}$, which is the last bit that we need to present a proof of Theorem 22.3.

Proof of Theorem 22.3. We take $g(\theta, \mathcal{D}) = N D_{\text{kl}}(\hat{\mathcal{L}}(\theta, \mathcal{D}) \| \mathcal{L}(\theta))$. Then:

$$\begin{aligned} \mathbb{P}\left(\exists \rho : D_{\text{kl}}(\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})] \| \mathbb{E}_\rho[\mathcal{L}(\theta)]) \geq \frac{D_{\text{KL}}(\rho \| \pi) + \ln \frac{2\sqrt{N}}{\delta}}{N}\right) \\ \leq \mathbb{P}\left(\exists \rho : \mathbb{E}_\rho[D_{\text{kl}}(\hat{\mathcal{L}}(\theta, \mathcal{D}) \| \mathcal{L}(\theta))] \geq \frac{D_{\text{KL}}(\rho \| \pi) + \ln \frac{2\sqrt{N}}{\delta}}{N}\right) \end{aligned} \quad (22.26)$$

$$= \mathbb{P}\left(\exists \rho : \mathbb{E}_\rho[N D_{\text{kl}}(\hat{\mathcal{L}}(\theta, \mathcal{D}) \| \mathcal{L}(\theta))] \geq D_{\text{KL}}(\rho \| \pi) + \ln \frac{2\sqrt{N}}{\delta}\right) \quad (22.27)$$

$$\leq \mathbb{P}\left(\exists \rho : \mathbb{E}_\rho[g(\theta, \mathcal{D})] \geq D_{\text{KL}}(\rho \| \pi) + \ln \frac{\mathbb{E}_\pi[\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}]]}{\delta}\right) \quad (22.28)$$

$$\leq \delta. \quad (22.29)$$

The first line of the derivation exploits convexity of D_{kl} and Jensen's inequality, which allows us to take the expectation outside of D_{kl} on the left side of the inequality inside the probability clause. The second line multiplies both sides of the inequality by N . The third line applies the kl lemma (22.4), by which $\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}] \leq 2\sqrt{N}$, and so $\mathbb{E}_\pi[\mathbb{E}_\mathcal{D}[e^{g(\theta, \mathcal{D})}]] \leq 2\sqrt{N}$. The last line is by the PAC-Bayes lemma (22.3).

We note that the requirement of boundedness of ℓ in the $[0, 1]$ interval is required in Lemma 22.4 and the requirement on independence of π and $\mathcal{D}$ is required in Lemma 22.3. $\square$

22.2 Ensemble classifiers and weighted majority vote

An alternative to randomised classification from Definition 22.1 is to take a ρ -weighted majority vote of the predictions given by parameters $\theta \in \Theta$. This approach has two advantages. One is that it transforms a randomised prediction into a deterministic one. (Imagine hearing that a doctor has sampled parameters of a model to predict your diagnosis. While it may be a statistically sound approach, psychologically it may be unsettling to some people.) The second, more important one, is the *cancellation of errors effect*. Informally, if in a set of classification models the individual models err with probability smaller than $\frac{1}{2}$ and their errors are independent, the errors tend to cancel out. For example, consider binary classification and imagine that we have a finite set $\Theta = (\theta_1, \dots, \theta_M)$ of size M , and that for every $\theta_i \in \Theta$ the probability of making an error on a random sample (x, y) is $\frac{1}{2} - \varepsilon$ for a small $\varepsilon > 0$, which implies that $\mathcal{L}(\theta) = \frac{1}{2} - \varepsilon$. Also, assume that for $i \neq j$ the predictions $f(x, \theta_i)$ and $f(x, \theta_j)$ are independent. Then a uniformly weighted majority vote of $f(x, \theta_1), \dots, f(x, \theta_M)$ will err only if at least half of the predictions are incorrect. By Hoeffding's inequality [761] the probability of this happening is $\mathbb{P}\left(\sum_{i=1}^M \mathbb{1}(f(x, \theta_i) \neq y) \geq \frac{1}{2}M\right) \leq e^{-2M\varepsilon^2}$, meaning that even for a small M the error rate of a uniformly weighted majority vote will be much smaller than the $\frac{1}{2} - \varepsilon$ error rate of the individual classifiers.

In practice, the true error rates are unknown and are not guaranteed to be below $\frac{1}{2}$, and the errors are not necessarily independent. If the error rates of individual models are above $\frac{1}{2}$, taking a majority vote may deteriorate the prediction accuracy. It is possible to use PAC-Bayesian analysis to provide generalisation bounds for weighted majority votes, as well as to optimise the weighting of classifiers ρ , as we demonstrate next.

From this point on we restrict ourselves to multiclass classification, i.e., $\mathcal{Y}$ is finite. We also assume that $f(x, \theta) \in \mathcal{Y}$. Under these assumptions, prediction of ρ -weighted majority vote $\text{MV}_\rho(x)$ is defined as

$$\text{MV}_\rho(x) = \arg \max_{y \in \mathcal{Y}} \mathbb{E}_{\theta \sim \rho} [\mathbb{1}(f(x, \theta) = y)], \quad (22.30)$$

where ties are resolved arbitrarily. In words, $\text{MV}_\rho(x)$ predicts the label that received the most ρ -weighted votes from the models $\theta \in \Theta$. In binary classification with $\mathcal{Y} = \{\pm 1\}$ and $f(x, \theta) \in \{\pm 1\}$, equation (22.30) can be simplified to

$$\text{MV}_\rho(x) = \text{sign}(\mathbb{E}_\rho[f(x, \theta)]). \quad (22.31)$$

22.3 Oracle bounds for the weighted majority vote

A key observation is that if a ρ -weighted majority vote made an error, we know that at least a ρ -weighted half of the predictors θ have erred. For the zero-one loss it implies

$$\ell(\text{MV}_\rho(x), y) \leq \mathbb{1}(\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)] \geq 0.5). \quad (22.32)$$

The observation leads to the following basic bound on the expected loss of the weighted majority vote.

Lemma 22.5 (A basic bound on the expected loss of a weighted majority vote).

$$\mathcal{L}(\text{MV}_\rho) \leq \mathbb{P}(\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)] \geq 0.5). \quad (22.33)$$

Proof. We have

$$\mathcal{L}(\text{MV}_\rho) = \mathbb{E}_{(x, y) \sim p(x, y)} [\ell(\text{MV}_\rho(x), y)] \quad (22.34)$$

$$\leq \mathbb{E}_{(x, y) \sim p(x, y)} [\mathbb{1}(\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)] \geq 0.5)] \quad (22.35)$$

$$= \mathbb{P}(\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)] \geq 0.5), \quad (22.36)$$

where the first line is by the definition of $\mathcal{L}(\cdot)$, the second line is by equation (22.32), and the last holds since $\mathbb{1}(\cdot)$ is a Bernoulli random variable. $\square$

The basic bound leads to a variety of oracle bounds, which are all based on taking a random variable $Z = \mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)]$ and bounding the probability $\mathbb{P}(Z \geq 0.5)$ on the right hand side of equation (22.33). At large, there is a “first order oracle bound”, which is based on the first moment of Z , and several variations of “second order oracle bounds”, which are based on the second moment of Z . It is possible, but uncommon to use higher moments of Z due to the statistical and computational difficulty of empirical estimation of higher moments.

First order oracle bound

The first bound is so basic that it was called “a folk theorem” [973].

Theorem 22.7 (First order oracle bound).

$$\mathcal{L}(\text{MV}_\rho) \leq 2\mathbb{E}_\rho[\mathcal{L}(\theta)]. \quad (22.37)$$

Proof. By Lemma 22.5 and Markov’s inequality (22.6)

$$\mathcal{L}(\text{MV}_\rho) \leq \mathbb{P}(\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)] \geq 0.5) \quad (22.38)$$

$$\leq 2\mathbb{E}_{(x,y) \sim p(x,y)}[\mathbb{E}_\rho[\mathbb{1}(f(x, \theta) \neq y)]] \quad (22.39)$$

$$= 2\mathbb{E}_\rho[\mathbb{E}_{(x,y) \sim p(x,y)}[\mathbb{1}(f(x, \theta) \neq y)]] \quad (22.40)$$

$$= 2\mathbb{E}_\rho[\mathcal{L}(\theta)]. \quad (22.41)$$

□

The bound is called an *oracle* bound, because it relates the expected loss of the majority vote to the expected losses of individual classifiers, $\mathcal{L}(\theta)$, which are unknown quantities. In the following section, we use PAC-Bayes to convert oracle bounds into empirical bounds. However, a shortcoming of the first order bound is evident already at the oracle level. The first order oracle bound takes no interactions between predictions of different classifiers into account, whereas cancellation of errors is the major source of power of the majority vote. The minimum of the first order oracle bound is achieved by placing all the mass $\rho(\theta)$ on the best performing classifier $\theta^* = \arg \min_{\theta} \mathcal{L}(\theta)$, which “kills” the majority vote. Even though in practice the first order empirical bound presented in the next section is often tighter than higher order bounds, it has been shown that optimisation of the weighting ρ through minimisation of the first order empirical bound typically degrades the expected loss $\mathcal{L}(\text{MV}_\rho)$ of the majority vote [1142].

Second order oracle bound

A line of works has sought to overcome the shortcoming of the first order oracle bound by applying the Chebyshev-Cantelli inequality instead of Markov’s inequality to bound $\mathbb{P}(Z \geq 0.5)$ [609, 954, 982, 1428], but they have suffered from statistical and computational inefficiency due to oracle quantities appearing in the denominator of the Chebyshev-Cantelli inequality. A breakthrough was achieved by Masegosa et al. [1142], who used a second order Markov’s inequality.

Theorem 22.8 (Second order Markov’s inequality). *For a random variable Z and $\varepsilon > 0$*

$$\mathbb{P}(Z \geq \varepsilon) \leq \frac{\mathbb{E}[Z^2]}{\varepsilon^2}. \quad (22.42)$$

Proof.

$$\mathbb{P}(Z \geq \varepsilon) \leq \mathbb{P}(Z^2 \geq \varepsilon^2) \leq \frac{\mathbb{E}[Z^2]}{\varepsilon^2}, \quad (22.43)$$

where the last step is by Markov’s inequality. □

The second order oracle bound is defined in terms of tandem loss, which we define next.

Definition 22.3 (Tandem loss). *The tandem loss of θ and θ' on (x, y) is defined by*

$$\ell(f(x, \theta), f(x, \theta'), y) = \mathbb{1}(f(x, \theta) \neq y) \mathbb{1}(f(x, \theta') \neq y). \quad (22.44)$$

The tandem loss counts an error if both θ and θ' erred on (x, y) . The expected tandem loss is defined by

$$\mathcal{L}(\theta, \theta') = \mathbb{E}_{(x,y) \sim p(x,y)}[\ell(f(x, \theta), f(x, \theta'), y)]. \quad (22.45)$$

And the empirical tandem loss is defined by

$$\hat{\mathcal{L}}(\theta, \theta', \mathcal{D}) = \frac{1}{N} \sum_{n=1}^N \ell(f(x_n, \theta), f(x_n, \theta'), y_n). \quad (22.46)$$

The tandem loss has the following property.

Lemma 22.6.

$$\mathbb{E}_{(x,y) \sim p(x,y)} \left[\left(\mathbb{E}_{\theta \sim \rho} [\mathbb{1}(f(x, \theta) \neq y)] \right)^2 \right] = \mathbb{E}_{(\theta, \theta') \sim \rho^2} [\mathcal{L}(\theta, \theta')], \quad (22.47)$$

where ρ^2 is a product distribution on $\Theta \times \Theta$.

The property relates the tandem loss to $\mathbb{E}_{\rho}[\mathbb{1}(f(x, \theta) \neq y)]$, which appears in the basic bound on the expected loss of weighted majority vote in Lemma 22.5 and will be used shortly. But before using it we provide a proof of the lemma.

Proof.

$$\mathbb{E}_{p(x,y)} \left[\mathbb{E}_{\rho} [\mathbb{1}(f(x, \theta) \neq y)]^2 \right] = \mathbb{E}_{p(x,y)} [\mathbb{E}_{\rho} [\mathbb{1}(f(x, \theta) \neq y)] \mathbb{E}_{\rho} [\mathbb{1}(f(x, \theta) \neq y)]] \quad (22.48)$$

$$= \mathbb{E}_{p(x,y)} [\mathbb{E}_{\rho^2} [\mathbb{1}(f(x, \theta) \neq y) \mathbb{1}(f(x, \theta') \neq y)]] \quad (22.49)$$

$$= \mathbb{E}_{\rho^2} [\mathbb{E}_{p(x,y)} [\mathbb{1}(f(x, \theta) \neq y) \mathbb{1}(f(x, \theta') \neq y)]] \quad (22.50)$$

$$= \mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')]. \quad (22.51)$$

□

Combining Lemma 22.5, Theorem 22.8, and Lemma 22.6 leads to the following result.

Theorem 22.9 (Second order oracle bound).

$$\mathcal{L}(\text{MV}_{\rho}) \leq 4 \mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')] \quad (22.52)$$

Proof. Take a random variable $Z = \mathbb{E}_{\rho}[\mathbb{1}(f(x, \theta) \neq y)]$ in Lemma 22.5, plug it into Theorem 22.8, and apply Lemma 22.6 to $\mathbb{E}[Z^2]$. □

A major advantage of the second order oracle bound over the first order oracle bound is that it takes correlations of errors into account through the tandem loss. Therefore, the weighting distribution ρ obtained by minimisation of the corresponding second order empirical bound typically does not deteriorate the expected loss of the resulting majority vote $\mathcal{L}(\text{MV}_{\rho})$ [1142]. A disadvantage is that in many cases the factor 4 in the bound is typically loose (typically $\text{MV}_{\rho}(x)$ performs better than the corresponding tandem classifier rather than within factor 4 worse) and as a bound (rather than as an optimisation objective) it is typically weaker than the first order bound.

Wu et al. [1780] proposed further refinement of the bound by replacing second order Markov's inequality with variational Chebyshev-Cantelli inequality.

Theorem 22.10 (Variational Chebyshev-Cantelli oracle bound). *For all ρ :*

$$\mathcal{L}(\text{MV}_{\rho}) \leq \min_{\mu < 0.5} \left(\frac{\mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')] - 2\mu \mathbb{E}_{\rho} [\mathcal{L}(\theta)] + \mu^2}{(0.5 - \mu)^2} \right). \quad (22.53)$$

We refer the reader to their paper for further details and an additional alternative form of the bound based on tandem losses with an offset. Additional improvements were proposed by Wu and Seldin [1779] and Wu et al. [1781].

For binary classification Germain et al. [609] and Masegosa et al. [1142] provide an alternative way of writing $\mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')]$, as a difference between $\mathbb{E}_{\rho} [\mathcal{L}(\theta)]$ and average disagreements between predictions. Since in binary classification disagreements can be estimated using unlabelled data only, this approach can benefit from large amounts of unlabelled data. However, for a noticeable advantage, the amount of unlabelled data should be more than square the amount of labelled data. We refer to Masegosa et al. [1142] for more details.

Comparison of the oracle bounds

We provide a brief comparison of the first and second order oracle bounds.

A positively correlated errors case. Since $\mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')] \leq \mathbb{E}_{\rho} [\mathcal{L}(\theta)]$, the second order oracle bound is at most twice worse than the first order oracle bound. The worst case occurs, for example, if all parameters θ in Θ give identical predictions. Then $\mathbb{E}_{\rho^2} [\mathcal{L}(\theta, \theta')] = \mathbb{E}_{\rho} [\mathcal{L}(\theta)] = \mathcal{L}(\text{MV}_{\rho})$ for all ρ and the first order bound has a factor 2 slack, whereas the second order bound has a factor 4 slack.

A negatively correlated errors case. Imagine that $|\Theta| = M$ is finite of size $M \geq 3$, such that each set of parameters $\theta \in \Theta$ errs on $1/M$ of the sample space (according to the distribution $p(x, y)$) and that the error regions are disjoint. Then $\mathcal{L}(\theta) = \mathcal{L}(\theta, \theta) = 1/M$ for all θ and $\mathcal{L}(\theta, \theta') = 0$ for all $\theta \neq \theta'$. For a uniform distribution ρ on Θ the first order bound is $2\mathbb{E}_\rho[\mathcal{L}(\theta)] = 2/M$ and the second order bound is $4\mathbb{E}_{\rho^2}[\mathcal{L}(\theta, \theta')] = 4/M^2$ and $\mathcal{L}(\text{MV}_\rho) = 0$. In this case the second order bound is an order of magnitude tighter than the first order.

An uncorrelated errors case. Assume that all parameter sets $\theta \in \Theta$ produce independent errors and have the same error rate, $\mathcal{L}(\theta) = \mathcal{L}(\theta')$ for all θ and θ' . Then for $\theta \neq \theta'$ we have $\mathcal{L}(\theta, \theta') = \mathbb{E}_{p(x, y)}[\mathbb{1}(f(x, \theta) \neq y)\mathbb{1}(f(x, \theta') \neq y)] = \mathbb{E}_{p(x, y)}[\mathbb{1}(f(x, \theta) \neq y)]\mathbb{E}_{p(x, y)}[\mathbb{1}(f(x, \theta') \neq y)] = \mathcal{L}(\theta)^2$ and $\mathcal{L}(\theta, \theta) = \mathcal{L}(\theta)$. For a uniform distribution ρ the second order bound is $4\mathbb{E}_{\rho^2}[\mathcal{L}(\theta, \theta')] = 4(\mathcal{L}(\theta)^2 + \frac{1}{M}\mathcal{L}(\theta)(1 - \mathcal{L}(\theta)))$ and the first order bound is $2\mathbb{E}_\rho[\mathcal{L}(\theta)] = 2\mathcal{L}(\theta)$. Assuming that M is large, so that we can ignore the second term in the second order bound, we obtain that it is tighter for $\mathbb{E}_\rho[\mathcal{L}(\theta)] < 1/2$ and looser otherwise. The former is the interesting regime.

22.4 Empirical bounds for the weighted majority vote

Empirical bounds for the weighted majority vote are obtained from the oracle bounds in a straightforward manner by applying any applicable PAC-Bayes bound to $\mathbb{E}_\rho[\mathcal{L}(\theta)]$ or $\mathbb{E}_{\rho^2}[\mathcal{L}(\theta, \theta')]$, depending on the order. Below we provide an example based on the optimisation-friendly form of the PAC-Bayesian bound in Theorem 22.2.

Theorem 22.11 (First order empirical bound). *Assume that ℓ is the zero-one loss function. For any prior distribution π over Θ that is independent of $\mathcal{D}$ and any $\delta \in (0, 1]$*

$$\mathbb{P}\left(\forall \rho : \mathcal{L}(\text{MV}_\rho) \leq 2 \min_{\lambda \in (0, 2)} \left(\frac{\mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]}{1 - \frac{\lambda}{2}} + \frac{\text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta}}{\lambda(1 - \frac{\lambda}{2})N} \right) \right) \geq 1 - \delta. \quad (22.54)$$

Theorem 22.12 (Second order empirical bound). *Assume that ℓ is the zero-one loss function. For any prior distribution π over Θ that is independent of $\mathcal{D}$ and any $\delta \in (0, 1]$*

$$\mathbb{P}\left(\forall \rho : \mathcal{L}(\text{MV}_\rho) \leq 4 \min_{\lambda \in (0, 2)} \left(\frac{\mathbb{E}_{\rho^2}[\hat{\mathcal{L}}(\theta, \theta', \mathcal{D})]}{1 - \frac{\lambda}{2}} + \frac{2 \text{D}_{\text{KL}}(\rho \parallel \pi) + \ln \frac{2\sqrt{N}}{\delta}}{\lambda(1 - \frac{\lambda}{2})N} \right) \right) \geq 1 - \delta. \quad (22.55)$$

Theorem 22.11 is obtained from Theorem 22.7 by applying Theorem 22.2 to $\mathbb{E}_\rho[\mathcal{L}(\theta)]$. Theorem 22.12 is obtained from Theorem 22.9 by applying Theorem 22.2 to $\mathbb{E}_{\rho^2}[\mathcal{L}(\theta, \theta')]$ and using the fact that $\text{D}_{\text{KL}}(\rho^2 \parallel \pi^2) = 2 \text{D}_{\text{KL}}(\rho \parallel \pi)$ (see Germain et al. [609, Page 814]).

We note that it is also possible to use Theorem 22.3 instead of Theorem 22.2 to obtain tighter bounds, but, as already mentioned, D_{KL} is less convenient for optimisation. A common practice is to use Theorem 22.11 or 22.12 to optimize ρ , and then to plug the optimised ρ into a D_{KL} -based bound to obtain a tighter generalisation certificate (which is a valid procedure, because Theorem 22.3 holds for all ρ , including ρ obtained by optimisation of any other bound).

Comparison of the empirical bounds

On top of the differences of the underlying oracle bounds discussed earlier, empirical bounds add some practical considerations. In general, $\mathbb{E}_{\rho^2}[\hat{\mathcal{L}}(\theta, \theta', \mathcal{D})] \leq \mathbb{E}_\rho[\hat{\mathcal{L}}(\theta, \mathcal{D})]$, thus the second order bound benefits from lower empirical risk and faster convergence rates (refer to the discussion of “fast convergence rates” in Theorem 22.1, where we mentioned that low empirical risk reduces the impact of the square root term). However, the second order bound has $2 \text{D}_{\text{KL}}(\rho \parallel \pi)$ in the complexity term compared to $\text{D}_{\text{KL}}(\rho \parallel \pi)$ in the first order bound. Additionally, in cross-validation and bagging-style applications it is data-efficient to use out-of-bag samples for the computation of the bounds. But since the tandem losses must be estimated on overlaps of the corresponding out-of-bag samples, there are less data for their estimation compared to out-of-bag samples available for estimation of plain losses. Therefore, the effective sample size $\mathcal{D}$ in Theorem 22.12 is smaller than in Theorem 22.11. We discuss this way of ensemble and bound construction in more detail in the next section. Masegosa et al. [1142] show that empirically as an objective for optimising ρ the second order bound most often produces majority votes with lower expected risk $\mathcal{L}(\text{MV}_\rho)$ than the first order bound. However, the first order bound often produces tighter generalisation guarantees. A practical possibility is to use the second order bound to find a good ρ and then plug it into the first order bound to get a generalisation guarantee, because the first order bound holds for all posteriors, including those produced by other methods.

22.5 PAC-Bayesian deep ensembles

There are numerous ways to construct prediction ensembles and various options for the choice of ensemble members, such as decision stumps, decision trees, or even ensembles of heterogeneous classifiers. In this book, we focus on ensembles of deep neural networks, see Chapter 19 for an introduction.

Combining deep neural networks into an ensemble model is a common and old approach to improving generalisation performance [161, 649, 705]. The most basic way to create an ensemble of M neural networks is to repeat the training process M times using all training data and to combine the resulting networks uniformly [705, 962]. This procedure can be viewed as a simple ensembling, where random weight initialisations and stochastic training (randomly composed mini-batches, randomised augmentations) are assumed to generate sufficiently diverse ensemble members. Although this approach is often explicitly referred to as non-Bayesian (see, e.g., Lakshminarayanan et al. [962], Ovadia et al. [1274], Wenzel et al. [1742]), the resulting ensemble may also be interpreted as a Bayesian model average from an approximate posterior [453, 692, 1765], see Chapter 19.

Uniform weighting is an easy, but not necessarily the best way to combine predictions. Optimisation of the weights has the potential to yield higher accuracy, as already pointed out in early works on ensembles [936]. The weights can be optimised using prior information and empirical performance of ensemble members.

The idea of using PAC-Bayesian bounds as optimisation objectives for optimising the weighting ρ has been floating around for a long time, however, as discussed earlier and demonstrated empirically in Masegosa et al. [1142], the first order bounds are unsuitable for this purpose, because they completely ignore (un)correlation of predictions, which is the main source of power for the majority vote. Therefore, meaningful optimisation of ρ is only possible based on second or higher-order bounds.

22.5.1 Construction of an ensemble

While the bounds presented above apply equally to finite and infinite sets Θ , computationally it is easier to work with finite sets. Infinite sets require approximations or highly restricted parametric models allowing closed-form calculation of the relevant integrals. We opt for a finite set $\Theta = (\theta_1, \dots, \theta_M)$ of prediction models, and then optimise weighting of the models in an ensemble by minimising a PAC-Bayes bound. Thiemann et al. [1599] have noted as long as the empirical risk estimates are unbiased, $\mathbb{E}[\hat{\mathcal{L}}(\theta, \mathcal{D})] = \mathcal{L}(\theta)$, it is possible to use different (sub)sets of the data to compute the empirical risk of different models θ_i in PAC-Bayesian bounds. For example, in cross-validation or bagging [205] a data point $(x, y) \in \mathcal{D}$ can be used in training some models in Θ and computing the empirical risk of the models that have not used it in training. (Seldin [1485] provides several exercises playing around with this approach.) To formalise this idea we let $\mathcal{D}_i$ denote the set of data that were *not* used for training model θ_i . Then $\mathbb{E}[\hat{\mathcal{L}}(\theta_i, \mathcal{D}_i)] = \mathcal{L}(\theta_i)$. We further use $\mathcal{D}_{i,j} = \mathcal{D}_i \cap \mathcal{D}_j$ to denote the set of data that were *not* used for training of neither θ_i nor θ_j . Then $\mathbb{E}[\hat{\mathcal{L}}(\theta_i, \theta_j, \mathcal{D}_{i,j})] = \mathcal{L}(\theta_i, \theta_j)$. We let $K = \min_{i,j} |\mathcal{D}_i \cap \mathcal{D}_j|$ denote the minimal overlap size and $\hat{\mathcal{L}} \in \mathbb{R}^{M \times M}$ denote the matrix of empirical tandem losses with entries $\hat{\mathcal{L}}_{i,j} = \hat{\mathcal{L}}(\theta_i, \theta_j, \mathcal{D}_{i,j})$. Theorem 22.12 undergoes a slight adjustment to accommodate the use of individualised sets of data for computation of empirical risks.

Theorem 22.13 (Second order empirical bound with individualised empirical risk estimators). *Assume that ℓ is the zero-one loss function. Let $\pi = (\pi_1, \dots, \pi_M)^\top$ and $\rho = (\rho_1, \dots, \rho_M)^\top$ be prior and posterior distributions on $\{1, \dots, M\}$, where π is independent of $\{\mathcal{D}_1, \dots, \mathcal{D}_M\}$. Assume that θ_i is independent of $\mathcal{D}_i$. Then for any $\delta \in (0, 1]$*

$$\mathbb{P}\left(\forall \rho : \mathcal{L}(\text{MV}_\rho) \leq 4 \min_{\lambda \in (0,2)} \left(\frac{\rho^\top \hat{\mathcal{L}} \rho}{1 - \frac{\lambda}{2}} + \frac{2 \text{D}_{\text{KL}}(\rho \| \pi) + \ln \frac{2\sqrt{K}}{\delta}}{\lambda(1 - \frac{\lambda}{2})K} \right) \right) \geq 1 - \delta, \quad (22.56)$$

where K and $\hat{\mathcal{L}}$ are defined above.

In other words, with probability at least $1 - \delta$, for all ρ and all $\lambda \in (0, 2)$

$$\mathcal{L}(\text{MV}_\rho) \leq 4 \left(\frac{\rho^\top \hat{\mathcal{L}} \rho}{1 - \frac{\lambda}{2}} + \frac{2 \text{D}_{\text{KL}}(\rho \| \pi) + \ln \frac{2\sqrt{K}}{\delta}}{\lambda(1 - \frac{\lambda}{2})K} \right). \quad (22.57)$$

22.5.2 Optimising ensemble weighting through minimisation of a second-order PAC-Bayesian bound

Similar to Theorem 22.2, the bound in Theorem 22.13 (the right hand side of equation (22.57)) is convex in λ for a fixed ρ . If the $\mathcal{D}_{i,j}$ are identical for all (i, j) pairs, then the bound is also convex in ρ for a fixed λ , however, if the $\mathcal{D}_{i,j}$ differ, convexity in ρ is not guaranteed (see Masegosa et al. [1142]). We optimise weighting ρ for the majority vote by alternating minimisation of the right hand side of the bound in equation (22.57) with respect to λ and with respect to ρ . For λ we use the exact minimisation step given earlier in equation (22.6). Namely, for a fixed ρ we take

$$\lambda_\rho = \frac{2}{\sqrt{\frac{2K\rho^\top \hat{\mathcal{L}}\rho}{2\text{D}_{\text{KL}}(\rho\|\pi) + \ln \frac{2\sqrt{K}}{\delta}} + 1 + 1}}. \quad (22.58)$$

For a fixed λ , ρ is updated by taking a gradient descent step, where the gradient ∇ of the right hand side of equation (22.57) is given by

$$\nabla = \frac{4}{1 - \frac{\lambda}{2}} \left(2\hat{\mathcal{L}}\rho + \frac{2}{\lambda K} \left(1 + \ln \frac{\rho}{\pi} \right) \right), \quad (22.59)$$

where $\ln \frac{\rho}{\pi} = \left(\ln \frac{\rho_1}{\pi_1}, \dots, \ln \frac{\rho_M}{\pi_M} \right)^\top$.

In practice, we represent ρ by unconstrained parameters ρ' using a differentiable mapping from $\mathbb{R}^M$ to the probability simplex Δ_M and consider the derivative of the objective w.r.t. ρ' . This allows us to apply unconstrained optimisation. For example, in the experiments below, we use the softmax function and an Rprop variant for gradient-based unconstrained optimisation [810].

Algorithm 25 summarises the minimisation procedure. Since the optimisation problem is low-dimensional ($\rho \in \mathbb{R}^M$), the algorithm quickly converges to a local minimum. Theorem 22.13 provides a generalisation guarantee for the ensemble produced by Algorithm 25.

Algorithm 25: Alternating minimisation of the bound in Theorem 22.13

Input: Matrix of empirical tandem losses $\hat{\mathcal{L}}$, prior π , size K of the smallest set $\mathcal{D}_{i,j}$, stopping criterion parameter T_ρ

Output: optimised ρ and λ

$\rho \leftarrow \frac{1}{M} \mathbf{1}_M$ // $\mathbf{1}_M$ is the M -dimensional vector of all ones

// Update λ using current ρ

$\lambda \leftarrow 2 \left(\sqrt{\frac{2K\rho^\top \hat{\mathcal{L}}\rho}{2\text{D}_{\text{KL}}(\rho\|\pi) + \ln \frac{2\sqrt{K}}{\delta}} + 1 + 1} \right)^{-1}$

while the objective $\frac{\rho^\top \hat{\mathcal{L}}\rho}{1 - \frac{\lambda}{2}} + \frac{2\text{D}_{\text{KL}}(\rho\|\pi) + \ln \frac{2\sqrt{K}}{\delta}}{\lambda(1 - \frac{\lambda}{2})K}$ decreases **do**

 // Update ρ using current λ

repeat

$\nabla = \frac{4}{1 - \frac{\lambda}{2}} \left(2\hat{\mathcal{L}}\rho + \frac{2}{\lambda K} \left(1 + \ln \frac{\rho}{\pi} \right) \right)$

 Update ρ by an iteration of gradient descent using ∇

until the objective $\frac{\rho^\top \hat{\mathcal{L}}\rho}{1 - \frac{\lambda}{2}} + \frac{2\text{D}_{\text{KL}}(\rho\|\pi) + \ln \frac{2\sqrt{K}}{\delta}}{\lambda(1 - \frac{\lambda}{2})K}$ does not improve for T_ρ steps

 // Update λ using current ρ

$\lambda \leftarrow 2 \left(\sqrt{\frac{2K\rho^\top \hat{\mathcal{L}}\rho}{2\text{D}_{\text{KL}}(\rho\|\pi) + \ln \frac{2\sqrt{K}}{\delta}} + 1 + 1} \right)^{-1}$

22.5.3 Empirical example

To illustrate optimisation of ensemble weighting via minimisation of the bound in Theorem 22.13, the approach is applied to stacking CNN LSTMs (models combining Convolutional Neural Networks and Long Short-Term Memory) [1816] to solve the IMDB binary classification benchmark [1100].

The IMDB data are commonly used to evaluate deep ensembles. We adopted an experimental setup similar to Wenzel et al. [1742]. The standard training protocol for the IMDB dataset includes the use of a validation

dataset to mitigate overfitting. Neural networks are trained on the training dataset, and the model parameters that yield the lowest error on the validation dataset are selected as the final model. We used 20,000 training samples and 5,000 random validation samples. The test set included all 25,000 samples from the original test set.

We compared three ensembling strategies. For that, we trained N_{trials} networks for N_{epochs} epochs and computed the validation error after each epoch using the model parameters at that point of the training process, also referred to as a model checkpoint. In the first method, referred to as $f_{\text{uniform}}^{\text{val}}$, we took the model checkpoint from the epoch with the lowest validation error for each of the N_{trials} training trials, ending up with an ensemble of $M = N_{\text{trials}}$ networks. These were uniformly combined into a deep ensemble. In the second method, $f_{\text{uniform}}^{\text{all}}$, we took all the model checkpoints from all trials, resulting in $M = N_{\text{trials}} \times N_{\text{epochs}}$ models in the ensemble. We combined them uniformly, disregarding the validation set. In the third method, f_{ρ}^{all} , the weighting of the ensemble from the previous method was optimised using Algorithm 25 and the validation set. The results, averaged over five repetitions, are shown in Figure 22.1. The uniform ensemble of the best checkpoints according to the validation error ($f_{\text{uniform}}^{\text{val}}$) performed better than taking all checkpoints from each trial ($f_{\text{uniform}}^{\text{all}}$), as expected when utilising validation data for model selection. However, after optimising the weights of the uniform ensemble using Algorithm 25 (f_{ρ}^{all}), the resulting ensemble classifier showed the lowest test error, even lower than the uniform ensemble of the best checkpoints ($f_{\text{uniform}}^{\text{val}}$).

The example shows that using validation data for optimising a second-order PAC-Bayesian bound does not only provide a performance guarantee, but can also lead to a better performing ensemble.

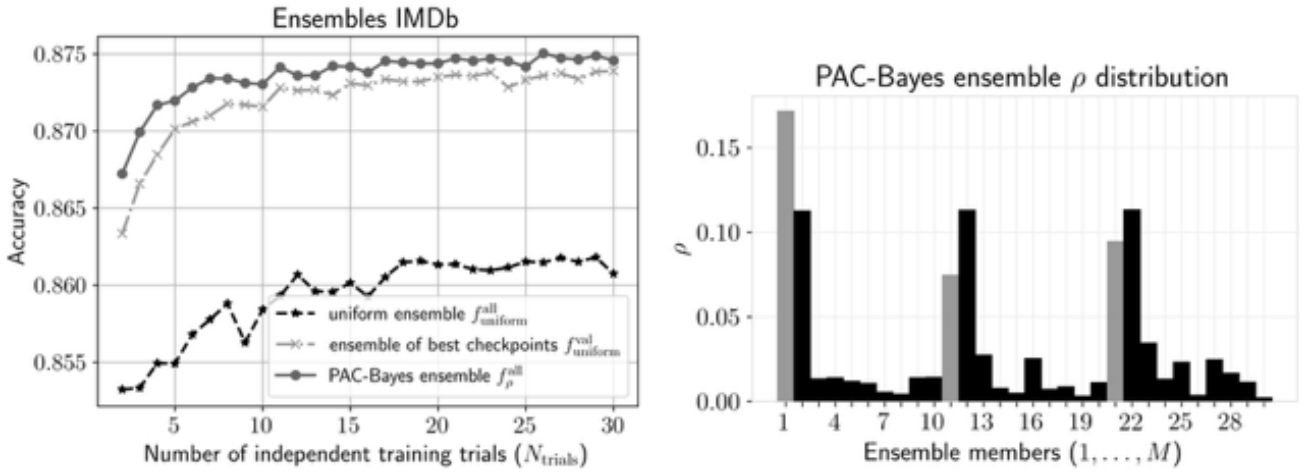


Figure 22.1: Left: Results for three different ensembling strategies for CNN LSTMs [1816] applied to the IMDb binary classification benchmark, averaged over five repetitions. We compared uniformly combining the best network from each trial as determined by the validation data set ($f_{\text{uniform}}^{\text{val}}$), combining uniformly the networks from all epochs and all trials ($f_{\text{uniform}}^{\text{all}}$), and a weighted combination of the networks from all epochs and all trials (f_{ρ}^{all}), where the weights were determined using Algorithm 25. Right: Distribution ρ over $N_{\text{trials}} = 3$ models with $N_{\text{epochs}} = 10$ checkpoints each (i.e., $M = 30$), the i -th snapshot from the j -th trial is assigned ensemble member number $i + N_{\text{epochs}} \cdot (j - 1)$. The figure shows that most weight is placed on the first two checkpoints of each trial, indicating overfitting after the early epochs. The lighter colour highlights the three checkpoints chosen by $f_{\text{uniform}}^{\text{val}}$, which are the initial checkpoints.

22.5.4 Ensemble pruning

The PAC-Bayesian bounds can further be used for ensemble pruning. After optimisation of ρ using a second order bound, pruning can be easily done by zeroing the lowest values of the optimised $\rho(\theta)$ and renormalising ρ . Since PAC-Bayesian bounds hold for all posterior distributions ρ , including those obtained by pruning, they provide a rigorous data-dependent trade-off between sparsity and accuracy of the pruned majority votes.

22.6 Summary

The chapter gave a basic introduction to PAC-Bayesian analysis, which is a frequentist way of incorporating prior information into learning. We contrasted PAC-Bayesian and Bayesian learning frameworks in Section 22.1.1.

One application of PAC-Bayesian analysis that we focused on is analysis and optimisation of weighted majority votes. It includes providing generalisation guarantees for weighted majority votes, optimising the weighting, and pruning.

Part V

Kernel methods

Chapter 23

Introduction to kernel methods

Kernel methods form one of the earliest and most elegant realizations of Bayesian learning in function space. Before deep networks, they provided a way to express flexible, non-parametric priors while preserving exact Bayesian inference, building on early work connecting Bayesian ideas to neural networks [412, 1226, 1612]. Today they play a dual role: (i) as stand-alone probabilistic models such as Gaussian Processes (GPs) and (ii) as analytic limits or building blocks of deep architectures—indeed, in the infinite-width limit, neural networks converge to GPs whose kernel is determined by the network architecture. From the perspective of **Bayesian deep learning**, kernels are the original tools for placing priors over functions rather than parameters.

While Parts I–IV developed inference schemes (sampling, variational, ensemble), Part V explores how the *function priors themselves* can be made expressive, deep, and learnable. The chapter supplies the fundamentals used throughout Part V and motivates the deep extensions that follow.

We begin with kernel functions and how they encode inductive biases (23.2), then introduce the *kernel trick*, the RKHS view, and the Representer theorem (23.3). Building on these foundations, we present Gaussian processes as distributions over functions and derive standard regression formulas (23.4). We then discuss the key limitations of standard kernels and GPs, including representation rigidity and scalability, and motivate deep, adaptive alternatives (23.5). This sets up the next chapters on Deep Gaussian Processes (25), Deep Kernel Learning (24), and Meta-Learning function priors (27).

23.1 Kernel as Feature Map

At the heart of kernel methods lies a powerful geometric intuition: complex patterns in raw data can often be understood more easily after transforming the data into a different representation space. A *kernel function* formalizes this idea by implicitly defining a transformation from an input space to a feature space where data relationships become more tractable.

23.1.1 From Input Space to Feature Space

Consider a dataset of objects $\{x_1, \dots, x_n\}$ residing in some *input space* $\mathcal{X}$. The input space represents the original, observed form of the data, for example, raw pixel values of an image, coordinates in Euclidean space, or even non-numeric structures like sequences or graphs.

A *feature map* $\phi : \mathcal{X} \rightarrow \mathcal{F}$ is a function that transforms each input x into a vector $\phi(x)$ in a *feature space* $\mathcal{F}$. The feature space is typically a vector space (often a Hilbert space) equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$, which allows us to measure angles, distances, and perform geometric operations.

The choice of feature map ϕ determines what aspects of the data are emphasized. By mapping data into a carefully chosen feature space, we can:

- Make nonlinear patterns in $\mathcal{X}$ appear linear in $\mathcal{F}$
- Encode domain knowledge about relevant data characteristics
- Change the dimensionality to enable different representations (increasing it for richer structure, or reducing it via feature selection)
- Define meaningful notions of similarity that are difficult to express in the original space

Algorithm 26: Kernel nearest neighbour regression

Input: kernel k on $\mathcal{X}$, minimum number of neighbours $n \in \mathbb{N}^+$, data $\{(x_i, y_i)\}_{i=1}^P \subset \mathcal{X} \times \mathbb{R}$, new input $x_* \in \mathcal{X}$

Output: predicted label y of x_*

$S = \{(x_i, y_i)\}_{i=1}^P$

$S_n = \emptyset$

while $|S_n| < n$ **do**

$S' \leftarrow \left\{ \operatorname{argmin}_{(x,y) \in S} \sqrt{k(x_*, x_*) - 2k(x_*, x) + k(x, x)} \right\}$

$S_n \leftarrow S_n \cup S'$

$S \leftarrow S \setminus S'$

return $y = \frac{1}{|S_n|} \sum_{(x_i, y_i) \in S_n} y_i$

23.1.2 Kernels as Inner Products in Feature Space

A *kernel function* $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ computes the inner product between feature representations without explicitly constructing those representations. Specifically, for any two inputs $x, x' \in \mathcal{X}$:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}} \quad (23.1)$$

That is, every kernel is a dot product in a (possibly infinite-dimensional) transformed space. This fact also holds in reverse: every feature map determines a kernel. This equation also reveals the kernel's dual nature: it is both a similarity measure in the input space and an inner product in the feature space. The kernel value $k(x, x')$ quantifies how aligned the feature representations $\phi(x)$ and $\phi(x')$ are in $\mathcal{F}$.

Geometric Interpretation. In the feature space $\mathcal{F}$, the inner product $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$ captures:

- **Magnitude and direction:** Large positive values indicate that $\phi(x)$ and $\phi(x')$ point in similar directions and have large norms
- **Similarity:** The inner product is maximized when the two vectors are parallel
- **Orthogonality:** When $k(x, x') = 0$, the feature representations are orthogonal, indicating complete dissimilarity in the feature space

Note that the inner product conflates magnitude and direction: a large value of $k(x, x')$ may reflect either strong alignment or large norms $\|\phi(x)\|$, $\|\phi(x')\|$. To interpret the inner product purely as directional similarity, one can normalize by the norms, yielding the cosine similarity $k(x, x') / \sqrt{k(x, x) k(x', x')}$. Alternatively, the kernel induces a metric in feature space:

$$d_k(x, x') = \|\phi(x) - \phi(x')\|_{\mathcal{F}} = \sqrt{k(x, x) + k(x', x') - 2k(x, x')}, \quad (23.2)$$

which provides a proper distance that separates magnitude from direction. The feature map can capture complex, nonlinear relationships, and this distance need not correspond to simple distance in the input space.

This metric immediately gives rise to simple kernelized algorithms. As an example, Algorithm 26 shows kernel nearest neighbour regression, which finds the closest training points to a query using only kernel evaluations and averages their labels.

23.1.3 Example: Quadratic Feature Map

To make these ideas concrete, consider a simple nonlinear transformation in two dimensions.

Example 23.1 (Quadratic Feature Map). *Let $\mathcal{X} = \mathbb{R}^2$ be the two-dimensional input space. Define the quadratic feature map $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ by:*

$$\phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad (23.3)$$

This mapping transforms each point $x = (x_1, x_2)^\top$ into a three-dimensional vector whose components are quadratic monomials of the input coordinates. The feature space $\mathcal{F} = \mathbb{R}^3$ is equipped with the standard Euclidean inner product.

Computing similarity via inner products. Consider two input points:

$$x = \begin{pmatrix} 5.9 \\ 3.0 \end{pmatrix}, \quad y = \begin{pmatrix} 6.9 \\ 3.1 \end{pmatrix}$$

Their feature representations are:

$$\phi(x) = \begin{pmatrix} 5.9^2 \\ 3.0^2 \\ \sqrt{2} \cdot 5.9 \cdot 3.0 \end{pmatrix} = \begin{pmatrix} 34.81 \\ 9.00 \\ 25.03 \end{pmatrix}$$

$$\phi(y) = \begin{pmatrix} 6.9^2 \\ 3.1^2 \\ \sqrt{2} \cdot 6.9 \cdot 3.1 \end{pmatrix} = \begin{pmatrix} 47.61 \\ 9.61 \\ 30.25 \end{pmatrix}$$

The inner product in feature space is:

$$\begin{aligned} \langle \phi(x), \phi(y) \rangle_{\mathcal{F}} &= 34.81 \times 47.61 + 9.00 \times 9.61 + 25.03 \times 30.25 \\ &= 1657.30 + 86.49 + 757.16 \\ &= 2500.95 \end{aligned}$$

This value quantifies the similarity between x and y in the quadratic feature space. Large positive values indicate strong similarity under this particular nonlinear transformation.

Why use feature maps? The quadratic feature map enables algorithms that operate via inner products (such as linear classifiers or regression) to capture quadratic decision boundaries in the input space. What appears as a nonlinear pattern in $\mathbb{R}^2$ becomes a linear pattern in $\mathbb{R}^3$.

23.1.4 The Computational Challenge

While feature maps provide a principled way to define similarity, they introduce a computational problem: the feature space $\mathcal{F}$ may be very high-dimensional, or even infinite-dimensional. Computing $\phi(x)$ explicitly can be expensive or impossible.

For instance, a polynomial feature map of degree q in d dimensions has feature space dimensionality:

$$\dim(\mathcal{F}) = \binom{d+q}{q}$$

which grows rapidly with both d and q . For $d = 100$ and $q = 3$, this yields over 176,000 dimensions. Explicitly computing and storing $\phi(x)$ for each data point becomes prohibitively expensive. Even more dramatically, some kernels, such as the Gaussian RBF kernel, correspond to infinite-dimensional feature spaces, making explicit feature computation impossible in principle.

This observation motivates the central insight of kernel methods, which we explore in Section 23.3: **if our algorithm can be expressed entirely in terms of inner products $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$, we need never compute $\phi(x)$ explicitly. Instead, we compute the kernel function $k(x, x')$ directly in the input space, bypassing the feature space entirely while achieving the same result.**

Remark 23.1. The kernel function $k(x, x')$ serves as an implicit representation of a potentially very complex feature map ϕ . This implicit representation is what makes kernel methods both computationally practical and theoretically elegant: we can work in rich, high-dimensional spaces without ever constructing them explicitly.

23.2 Kernel Functions

A *kernel function* provides a principled way to measure alignment between data points in a feature space, which can informally be thought of as the similarity between data points. Formally, a kernel is a symmetric function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that satisfies certain mathematical properties, i.e. symmetry and positive semi-definiteness of the corresponding kernel (or Gram) matrices. Given any $N \in \mathbb{N}$ and points $\{x_i\}_{i=1}^N$, the Gram matrix $K \in \mathbb{R}^{N \times N}$ with $K_{ij} = k(x_i, x_j)$ must be positive semidefinite.

23.2.1 Positive semi-definite kernels

Definition 23.1 (Positive semi-definite kernel). A function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a positive semi-definite (psd) kernel if it is symmetric, i.e., $k(x, x') = k(x', x)$ for all $x, x' \in \mathcal{X}$, and for any finite set of points $\{x_1, \dots, x_N\} \subset \mathcal{X}$ and any $\alpha \in \mathbb{R}^N$,

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j k(x_i, x_j) = \alpha^\top K \alpha \geq 0, \quad (23.4)$$

where $K \in \mathbb{R}^{N \times N}$ is the Gram matrix with entries $K_{ij} = k(x_i, x_j)$. A kernel is called strictly positive definite if the inequality is strict for all $\alpha \neq \mathbf{0}$ whenever the points $x_1, \dots, x_N$ are distinct.

In the literature, psd kernels are often simply called “positive definite kernels.” The positive semi-definiteness of K guarantees that the kernel corresponds to an inner product in some (possibly infinite-dimensional) feature space, enabling geometric reasoning about similarity, a connection we explore in detail in 23.3. Most commonly used kernels (e.g., the squared exponential) are in fact strictly positive definite, but the psd condition suffices for the theory developed here.

Common Examples

- **Linear kernel:** $k(x, x') = x^\top x'$; when used in GP regression, this is equivalent to Bayesian linear regression with a Gaussian prior.
- **Polynomial kernel:** $k(x, x') = (x^\top x' + c)^d$ with $c \geq 0$, $d \in \mathbb{N}$.
- **Squared Exponential (Gaussian/RBF):**

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right),$$

where the length-scale parameter $\ell > 0$ controls smoothness (larger ℓ means smoother functions).

- **Matérn kernel:** A family of kernels parameterized by smoothness $\nu > 0$:

$$k_\nu(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{r}{\ell}\right)^\nu K_\nu\left(\sqrt{2\nu} \frac{r}{\ell}\right)$$

where $r = \|x - x'\|$, $\ell > 0$ is the length-scale, and K_ν is the modified Bessel function of the second kind. Common special cases include Matérn-1/2, Matérn-3/2, and Matérn-5/2.

- **Laplacian Kernel:**

$$k(x, x') = \exp(-\gamma \|x - x'\|_1),$$

where $\|x - x'\|_1$ is the Manhattan distance (or L1 norm) between the vectors (the L2 norm can also be used) and γ is a tuning parameter that controls the width or “spread” of the kernel. A small γ gives a broad, smooth kernel, while a large γ creates a sharper, more localized kernel.

- **Indicator Kernel:**

$$k(x, x') = \mathbf{1}(x = x'),$$

where $\mathbf{1}(\cdot)$ is the indicator function.

- **Arc-cos kernel:** Used in neural network GP limits (26):

$$k(x, x') = \sigma_b^2 + \frac{\sigma_w^2}{\pi} \sqrt{k_{11}k_{22}} (\sin \theta + (\pi - \theta) \cos \theta)$$

where $\theta = \arccos(k_{12}/\sqrt{k_{11}k_{22}})$ with $k_{ij} = x_i^\top x_j / N_0$.

- **Periodic kernel:** $k(x, x') = \exp[-2 \sin^2(\pi \|x - x'\|/p)/\ell^2]$.

23.2.2 Kernel as a Similarity Measure

Having defined several kernel families above, we now make explicit the role that the kernel value plays as a notion of similarity. A kernel value $k(x, x')$ measures similarity between two inputs x and x' . In later sections where a function f is modelled with this kernel, $k(x, x')$ will play the role of the expected covariance between the function evaluations $f(x)$ and $f(x')$. Large values imply that function values at x and x' are likely similar, while small values indicate independence or dissimilarity. Hence, the kernel implicitly defines a notion of distance or correlation in the input space: the smaller the effective distance between x and x' , the stronger their correlation.

Different kernels correspond to different inductive biases about how similarity should behave:

- The RBF (squared exponential) kernel corresponds to very smooth, globally correlated functions that vary slowly with input distance.
- The Matérn family provides explicit control of smoothness and local variability through the parameter ν , enabling rougher or smoother function priors.
- Periodic kernels impose repeating structure, suitable for cyclic or seasonal patterns.

The practitioner's choice of kernel thus amounts to specifying a *prior* over function behaviour and determining what kinds of variation are expected in unseen regions of the input space.

23.2.3 Kernel Compositions and Operations

A powerful feature of kernels is that they can be combined to create more expressive similarity measures while preserving positive definiteness. Sums, products, and positive scalings of kernels preserve positive definiteness.

Proposition 23.1 (Kernel Operations). *If k_1 and k_2 are positive definite kernels, then so are:*

1. *Sum:* $k(x, x') = k_1(x, x') + k_2(x, x')$
2. *Product:* $k(x, x') = k_1(x, x') \cdot k_2(x, x')$
3. *Scaling:* $k(x, x') = c \cdot k_1(x, x')$ for $c > 0$

These composition rules enable practitioners to build complex kernels from simple building blocks, encoding structured prior knowledge. For example, a sum of periodic and SE kernels can model functions with both trend and seasonal components.

23.2.4 Spectral Perspective

This subsection can be skipped on a first reading.

For stationary kernels, i.e. those that depend only on the difference $\tau = x - x'$, it is often useful to describe them in terms of their frequency content rather than their shape in input space. *Bochner's theorem* [176] provides this connection: a continuous, shift-invariant kernel $k(\tau)$ on $\mathbb{R}^d$ is positive definite if and only if it is the Fourier transform of a nonnegative measure $S(\omega)$:

$$k(\tau) = \int_{\mathbb{R}^d} e^{i\omega^\top \tau} S(\omega) d\omega. \quad (23.5)$$

The function $S(\omega)$ is called the *spectral density* of the kernel and describes how much each frequency ω contributes to the covariance between inputs.

This spectral view provides an intuitive link between a kernel's smoothness and its frequency content:

- Smooth kernels (e.g., the RBF) have rapidly decaying $S(\omega)$, concentrating power at low frequencies and producing slowly varying functions.
- Rougher kernels (e.g., Matérn) have heavier-tailed $S(\omega)$, reflecting higher-frequency variability and less smoothness.

This spectral viewpoint motivates approximations like *random Fourier features*, which represent kernels through sampled frequencies $\omega \sim S(\omega)$. It also clarifies how kernel hyperparameters (e.g., the length-scale) determine the effective frequency range of the modelled functions.

Finally, some kernels are called *universal* or *characteristic*, meaning they can approximate any continuous function on compact domains or uniquely represent probability distributions through their mean embeddings. These properties make them powerful tools for both regression and statistical testing.

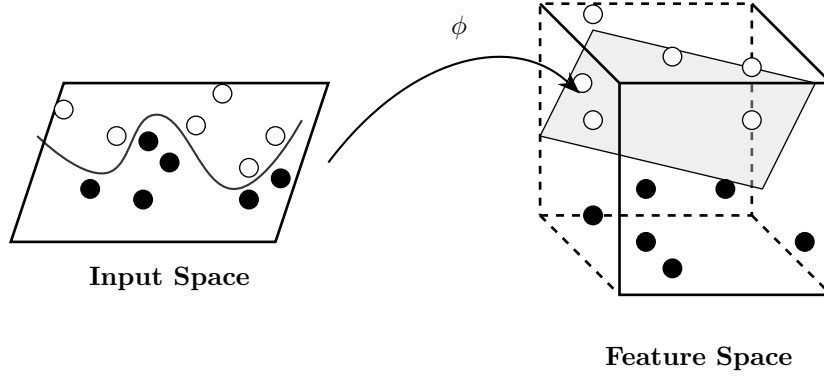


Figure 23.1: **The kernel trick.** A nonlinear mapping $\phi(x)$ transforms data from the input space (left) to a high-dimensional feature space (right) where the classes become linearly separable. Although the feature space may be high- or infinite-dimensional, the kernel trick allows computation using only pairwise kernel evaluations $k(x, x') = \langle \phi(x), \phi(x') \rangle$, without explicitly constructing $\phi(x)$.

23.3 The Kernel Trick

The kernel trick is a computational technique that enables efficient operations in high-dimensional (even infinite-dimensional) feature spaces without explicitly computing the features themselves. This elegant idea underpins much of kernel methods' success.

23.3.1 Feature Maps and Implicit Representations

Every positive definite kernel $k(\cdot, \cdot)$ corresponds to a feature map $\phi : \mathcal{X} \rightarrow \mathcal{H}$ into some Hilbert space $\mathcal{H}$ such that:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \quad (23.6)$$

The key insight is that we can perform all computations using only kernel evaluations, without ever explicitly computing $\phi(x)$. In other words, Equation (23.6) shows that every positive definite kernel implicitly defines a mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that the kernel value is simply the inner product between features in $\mathcal{H}$. The feature space $\mathcal{H}$ can be extremely high-dimensional or even infinite, but all computations depend only on the pairwise kernel evaluations $k(x, x')$.

This observation means that any algorithm that can be expressed in terms of dot products can be *kernelized* by substituting $\langle x, x' \rangle \rightarrow k(x, x')$. This single substitution turns linear methods into nonlinear ones while preserving the same algebraic structure.

As illustrated by the quadratic feature map in Example 23.1, computing $\phi(x)$ explicitly for N_0 -dimensional input requires $O(N_0^2)$ operations, but evaluating the kernel directly takes only $O(N_0)$ operations. The kernel trick thus achieves the same geometric effect of projecting into a different feature space at a fraction of the computational cost. For some kernels like the squared exponential (SE), the corresponding feature space is infinite-dimensional, making explicit feature computation impossible. Yet the kernel can be evaluated efficiently in $O(N_0)$ time.

23.3.2 Reproducing Kernel Hilbert Spaces

The mathematical foundation underlying the kernel trick is the concept of a *Reproducing Kernel Hilbert Space* (RKHS). An RKHS provides the link between kernel functions and inner product spaces of functions, ensuring that evaluation at any point can be represented as an inner product.

Definition 23.2 (Reproducing Kernel Hilbert Space). *Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$. We say that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space if there exists a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ satisfying:*

1. *For every $x \in \mathcal{X}$, the function $k(x, \cdot)$ belongs to $\mathcal{H}$.*
2. (**Reproducing property**) *For all $f \in \mathcal{H}$ and all $x \in \mathcal{X}$,*

$$f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}. \quad (23.7)$$

The reproducing property ensures that pointwise evaluation is a continuous linear functional in $\mathcal{H}$, a feature that distinguishes RKHSs from generic Hilbert spaces of functions. Each positive definite kernel $k(\cdot, \cdot)$ uniquely defines an RKHS $\mathcal{H}_k$, and conversely, every RKHS admits a unique reproducing kernel. This result is known as the **Moore-Aronszajn Theorem**, cf. e.g. [128].

This duality establishes a one-to-one correspondence between kernels and function spaces: the choice of kernel implicitly determines the class of admissible functions and their smoothness. For instance, the RKHS associated with the SE kernel consists of infinitely differentiable functions, while that of the Matérn kernel contains functions with finite smoothness controlled by the parameter ν .

23.3.3 Computational Advantage and the Representer Theorem

The computational power of the kernel trick becomes evident in regularized learning problems defined over a Reproducing Kernel Hilbert Space $\mathcal{H}$. Given a training set $\{(x_i, y_i)\}_{i=1}^P \subset \mathcal{X} \times \mathbb{R}$, consider the objective

$$\min_{f \in \mathcal{H}} \sum_{i=1}^P \ell(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2, \quad (23.8)$$

where $\ell(\cdot, \cdot)$ is a pointwise loss function (e.g., squared, hinge, or logistic loss) and $\lambda > 0$ is a regularization parameter controlling model complexity.

The **Representer Theorem** [891, 1472] states that any optimal solution to (23.8) can be expressed as a finite linear combination of kernel evaluations at the training points:

$$f^*(x) = \sum_{i=1}^P \alpha_i k(x_i, x), \quad (23.9)$$

for some coefficients $\alpha = [\alpha_1, \dots, \alpha_P]^\top \in \mathbb{R}^P$. This result implies that even though $\mathcal{H}$ may be infinite-dimensional, the optimization problem reduces to one over the coefficients α , involving only the kernel (Gram) matrix K with entries $K_{ij} = k(x_i, x_j)$. Predictions thus require only kernel evaluations $k(x_i, x)$ rather than explicit feature computations, providing the foundation for the efficiency of kernel methods.

23.4 Gaussian Processes

Having introduced kernels as measures of similarity and as inner products in feature spaces, we are now ready to use them as *covariance functions* that define distributions over functions. A Gaussian process (GP) can be thought of as the probabilistic analogue of a kernel machine: where a kernel specifies how inputs are related, a GP interprets this as a statement about how function values co-vary across the input space. In this sense, GPs turn the geometry induced by a kernel into a fully Bayesian model of function learning. Frequentist kernel methods and GPs are closely related and share common mathematical foundations, as detailed in [865].

GP extends the notion of a Gaussian distribution from finite-dimensional vectors to functions (thinking of functions as infinitely long vectors containing the value of the function at every input).

Definition 23.3 (Gaussian Process [1382]). *Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a real-valued function defined on an input space $\mathcal{X}$. We say that f is a Gaussian Process, denoted*

$$f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)),$$

if for any finite collection of inputs $\mathbf{x} = [x_1, \dots, x_P]^\top$, the random vector of function evaluations

$$\mathbf{f} = [f(x_1), \dots, f(x_P)]^\top$$

follows a multivariate normal distribution

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, K_{\mathbf{xx}}),$$

with mean $\boldsymbol{\mu}_i = m(x_i)$ and covariance $[K_{\mathbf{xx}}]_{ij} = k(x_i, x_j)$.

The mean function $m(x) = \mathbb{E}[f(x)]$ describes the expected value of the process at x , while the kernel or covariance function $k(x, x') = \text{Cov}[f(x), f(x')]$ encodes how function values co-vary across inputs. In practice, one often assumes $m(x) = 0$. This is reasonable when the data have been centred or when no specific prior knowledge about the function's mean behaviour is available; the mean function can always be absorbed into the model by subtracting a prior estimate of the trend from the observations before fitting the GP.

Kernel Matrix. For inputs $\mathbf{x} = [x_1, \dots, x_P]$, the kernel matrix is

$$K_{\mathbf{xx}} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_P) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_P) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_P, x_1) & k(x_P, x_2) & \cdots & k(x_P, x_P) \end{bmatrix}.$$

As established in Definition 23.1, the kernel must be symmetric and positive semi-definite, which guarantees that $K_{\mathbf{xx}}$ defines a valid covariance matrix.

Remark 23.2. *The kernel function in a GP both defines the covariance structure of the process and determines the similarity measure for interpolation. This connection between the probabilistic and geometric views is fundamental to GP reasoning.*

23.4.1 Posterior Inference

Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^P$, we assume that observations are noisy evaluations of an underlying latent function f :

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

where σ^2 denotes the noise variance.

Let $\mathbf{y} = [y_1, \dots, y_P]^\top$ be the observed targets, and consider a new input x_* with latent value $f_* = f(x_*)$. The joint prior over training and test outputs is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K_{\mathbf{xx}} + \sigma^2 I & K_{\mathbf{x}x_*} \\ K_{x_*\mathbf{x}} & K_{x_*x_*} \end{bmatrix}\right),$$

where $K_{\mathbf{x}x_*} = [k(x_1, x_*), \dots, k(x_P, x_*)]^\top$. The noise term $\sigma^2 I$ is added only to $K_{\mathbf{xx}}$ because the training targets $\mathbf{y}$ are noisy observations, whereas f_* denotes the latent (noise-free) function value at x_* .

Conditioning on the observed data gives the posterior distribution:

$$p(f_* | \mathbf{x}, \mathbf{y}, x_*) = \mathcal{N}(\mu_*, \sigma_*^2),$$

with

$$\mu_* = K_{x_*\mathbf{x}} [K_{\mathbf{xx}} + \sigma^2 I]^{-1} \mathbf{y}, \quad \sigma_*^2 = k(x_*, x_*) - K_{x_*\mathbf{x}} [K_{\mathbf{xx}} + \sigma^2 I]^{-1} K_{\mathbf{x}x_*}.$$

Interpretation. The posterior mean μ_* provides the predictive expectation of $f(x_*)$, while the variance σ_*^2 quantifies uncertainty. Near observed data points, σ_*^2 tends to be small, while it increases in regions far from the training data. This property makes GPs particularly useful for uncertainty-aware regression, Bayesian optimization, and active learning (see Chapter 53).

23.5 Limitations

Despite their elegance, standard GPs face several fundamental challenges that motivate the deep extensions explored in Chapters 25–27. These limitations concern both their *representational capacity* and their *computational scalability*.

23.5.1 Fixed Representations and Expressiveness

Standard kernel methods operate directly on raw input features, defining similarity through fixed metrics such as the Euclidean distance. However, this assumption often fails in modern high-dimensional or structured data domains.

Uninformative raw features. In domains such as images, text, or graphs, the raw input coordinates lack semantic content and do not encode the relational or compositional structure necessary for generalization. Kernels defined over such features therefore fail to capture the meaningful invariances or dependencies that underlie the data.

High-dimensional inputs. As the dimensionality of the input space grows, pairwise distances tend to concentrate, making commonly used stationary kernels, such as the RBF, nearly constant. This “distance concentration” effect diminishes the discriminative power of kernel similarity measures, rendering them ineffective in capturing local geometry or fine-grained variation within the data manifold.

Limited adaptability. Once the functional form and hyperparameters of a kernel are fixed, its inductive bias becomes static. Consequently, standard kernels cannot learn task- or data-dependent notions of similarity, preventing them from adapting to complex, structured patterns. This lack of flexibility fundamentally restricts their expressiveness in modern representation learning settings, where adaptive, data-driven feature extraction is essential.

23.5.2 Computational Scalability

For most practical applications, exact Gaussian process inference is computationally expensive: evaluating and inverting the $P \times P$ kernel matrix scales as $O(P^3)$, which limits standard GPs to datasets of a few thousand points.

To make GPs tractable for large datasets and complex likelihoods, approximate inference schemes are employed:

- **Variational inference:** introduces an approximate posterior distribution that can be optimized efficiently and supports non-Gaussian likelihoods (e.g., classification or count models) [1615].
- **Sparse and inducing-point methods:** summarize the dataset using a small set of representative inputs, reducing complexity to $O(P_i^3 + P_i^2 P)$ with $P_i \ll P$ [1764].

These ideas are central to the deep kernel models discussed later. Each layer or GP component uses such approximations to remain scalable. Scalable GP methods are treated in depth in Chapter 38, and detailed derivations of deep kernel models appear in Chapters 24 and 27.

23.5.3 The Path Forward: Deep and Adaptive Gaussian Processes

The next chapters explore how to overcome these limitations by making *the kernel itself learnable, hierarchical, and data-adaptive*. This evolution parallels the transition from linear models to deep networks in standard machine learning.

- **Chapter 25 — Deep Gaussian Processes:** Introduces hierarchical compositions of GPs, where the outputs of one GP feed into another. These deep GPs capture multiple levels of abstraction and non-stationarity, addressing the expressiveness gap of shallow kernels.
- **Chapter 24 — Deep Kernel Learning:** Combines the representation power of deep neural networks with the uncertainty quantification of GPs. Here, the kernel is learned from data via a neural network feature extractor, providing scalable, end-to-end trainable models.
- **Chapter 27 — Meta-Learning and Function Priors:** Generalizes the concept further by learning priors over functions across tasks, enabling few-shot adaptation and transfer. These models tie together the Bayesian principles of GPs with the flexibility of meta-learning.

Together, these developments trace the progression from classical kernel methods to modern *Bayesian deep learning in function space*. Each subsequent chapter builds directly on the probabilistic foundations established here, transforming GPs from elegant theoretical constructs into scalable, representation-learning engines.

BDL Connection

How kernel methods connect to Bayesian deep learning.

Kernel methods are among the earliest approaches to Bayesian learning in function space [1382]. A *kernel* defines a covariance structure over functions, specifying how values at different inputs are expected to co-vary. In Gaussian processes, this covariance is combined with a Gaussian prior to produce full predictive distributions, capturing both predictions and their uncertainty.

From the perspective of deep learning, kernel methods provide the *Bayesian limit of neural networks*: when a network's width tends to infinity, its outputs converge to a GP whose kernel is determined by the network architecture (e.g. the Neural Network Gaussian Process [1000, 1226, 1759] or Neural Tangent Kernel [829, 1794]). Thus, every deep network implicitly defines a kernel and training it corresponds to learning that kernel's geometry.

Modern *Bayesian deep learning* builds on this connection:

- **Representation learning:** Deep kernel learning replaces the fixed input kernel by one computed from a learned deep representation.

- **Hierarchical function priors:** Deep Gaussian processes stack multiple GPs to model complex, non-stationary mappings.
- **Uncertainty calibration:** The probabilistic nature of GPs provides well-founded uncertainty estimates, a property extended in BDL models.

Understanding kernels therefore clarifies the foundations on which deep Bayesian models are built: they describe *how we believe functions should behave* before data are seen, and how these beliefs evolve as models grow deeper.

Chapter 24

Deep kernel learning

Historically, neural networks and kernel methods have often been treated as competing approaches. Deep kernel learning (DKL) [1769] combines a deterministic neural-network feature extractor with a kernel method, most commonly a Gaussian process (GP). At prediction time, standard DKL evaluates the feature extractor once for each test input and then performs GP prediction in the learned feature space. This avoids the repeated stochastic neural-network passes required by some Bayesian neural-network approximations, but GP prediction still requires kernel computations against stored training or inducing-point features.

Deep kernel learning transforms the inputs of a base kernel, such as a radial basis function (RBF) kernel, through a neural network. The resulting *deep kernel* has hyperparameters from both the base kernel and the neural-network feature map. For Gaussian-likelihood regression, the GP predictive distribution and marginal likelihood are available in closed form for any fixed values of these hyperparameters. Learning the hyperparameters still requires numerical optimisation, and classification or other non-Gaussian likelihoods require approximate inference. Moreover, standard DKL is an empirical-Bayes (type-II) construction: it integrates over GP function values but point-estimates the neural-network and kernel parameters rather than placing a posterior over them.

The remainder of the chapter reviews Gaussian processes (Section 24.1), introduces the deep-kernel construction (Section 24.2), describes stochastic variational deep kernel learning (Section 24.3), discusses estimation choices (Section 24.4), and closes with practical resources (Section 24.5).

Applications and surveys involving DKL or closely related GP–neural-network hybrids span biological sequence design [1557], physics-informed machine learning [872], materials science [299], computational chemistry [877], autonomous vehicles [18], semi-supervised learning [838], few-shot learning [1298], healthcare [268, 1034], and astrobiology [1880]. Related hybrid methods and DKL extensions include [202, 226, 1062, 1714]. DKL has also been used in Bayesian optimisation [e.g., 1036, 1557], where predictive uncertainty informs the exploration–exploitation trade-off; what counts as “far from the data”, however, is determined by the learned representation.

24.1 Review of Gaussian Processes

Although deep kernels can be used with other kernel methods, this chapter focuses on Gaussian processes because they provide both predictive distributions and a marginal-likelihood objective for hyperparameter estimation.

For a full treatment, see Rasmussen and Williams [1382]. A *Gaussian process* is a distribution over functions whose every finite-dimensional marginal is Gaussian. We write $f \sim \mathcal{GP}(m, k)$ when, for every finite input set $X = \{x_1, \dots, x_n\}$,

$$(f(x_1), \dots, f(x_n))^{\top} \sim \mathcal{N}(m_X, K_{XX}), \quad (24.1)$$

where $(m_X)_i = m(x_i)$ and $(K_{XX})_{ij} = k(x_i, x_j)$.

For simplicity, we assume $m(x) = 0$ unless stated otherwise. The kernel then plays the central role in determining the covariance and generalisation properties of the process. A standard choice is the RBF kernel

$$k_{\text{RBF}}(x_i, x_j) = a^2 \exp\left(-\frac{\|x_i - x_j\|^2}{2\ell^2}\right), \quad (24.2)$$

where the amplitude a^2 controls the signal variance and the length-scale ℓ controls how rapidly correlations decay with distance. Larger length-scales produce more slowly varying functions, whereas shorter length-scales permit more rapidly varying functions. In forecasting, for example, ℓ can be interpreted as the time range over which past observations remain substantially correlated with a forecast.

For Gaussian-likelihood regression, assume

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2),$$

with the noise independent of f . Given training inputs $X = \{x_i\}_{i=1}^N$ and observations $\mathbf{y} = (y_1, \dots, y_N)^\top$, the latent-function predictive distribution at x_* is

$$\begin{aligned} p(f_* | \mathbf{y}, x_*, X) &= \mathcal{N}(f_* | \mu_*, v_*), \\ \mu_* &= k_{*X} (K_{XX} + \sigma^2 \mathbf{I}_N)^{-1} \mathbf{y}, \\ v_* &= k_{**} - k_{*X} (K_{XX} + \sigma^2 \mathbf{I}_N)^{-1} k_{X*}. \end{aligned} \quad (24.3)$$

The predictive distribution for a noisy observation y_* has the same mean and variance $v_* + \sigma^2$.

Writing $\mathbf{f} = (f(x_1), \dots, f(x_N))^\top$, the marginal likelihood is

$$p(\mathbf{y} | \theta, \sigma^2, X) = \int p(\mathbf{y} | \mathbf{f}, \sigma^2) p(\mathbf{f} | \theta, X) d\mathbf{f}. \quad (24.4)$$

For Gaussian-likelihood regression,

$$\log p(\mathbf{y} | \theta, \sigma^2, X) = -\frac{1}{2} \mathbf{y}^\top (K_\theta + \sigma^2 \mathbf{I}_N)^{-1} \mathbf{y} - \frac{1}{2} \log |K_\theta + \sigma^2 \mathbf{I}_N| + \text{const}. \quad (24.5)$$

The first term measures data fit, while the log-determinant term acts as a complexity penalty and connects to minimum description length [1111]. The marginal likelihood can therefore be used to estimate kernel hyperparameters and the observation-noise variance.

24.2 Deep Kernel Learning

Let k_θ be a base kernel with hyperparameters θ , and let g_w be a neural-network feature map with weights w . The resulting deep kernel is

$$k_\gamma(x, x') := k_\theta(g_w(x), g_w(x')), \quad \gamma = (\theta, w). \quad (24.6)$$

Composition with a deterministic feature map preserves positive semidefiniteness, so k_γ is a valid kernel whenever k_θ is. Structurally, this places a GP on the features produced by the neural network.

The base kernel supplies a covariance model, while the neural network learns a task-adapted representation in which that covariance model is applied. For Gaussian-likelihood regression and fixed γ , prediction requires one forward pass through g_w for each test input, followed by GP prediction using cross-covariances with stored training or inducing-point features. When γ is learnt by marginal-likelihood maximisation, the usual DKL construction is empirical Bayes: w and θ are point-estimated rather than integrated out.

Once constructed, k_γ can be used like any other kernel. The principal estimation strategies are discussed in Section 24.4.

Dense GP regression requires $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory. Structured kernel interpolation (SKI, also known as KISS-GP) [1764] introduces a structured inducing grid and sparse local interpolation. In favourable regimes, this yields fast matrix-vector products and iterative inference whose cost can be close to linear in N , with additional dependence on the number and structure of the inducing points. SKI is an approximation rather than exact GP inference, and it is most effective in low-dimensional spaces; a neural feature map can supply such a space.

Other scalable GP approaches can also be combined with DKL. GPU-accelerated conjugate-gradient solves and stochastic log-determinant estimates are effective in practice [580], but finite solve tolerances and stochastic estimators remain numerical approximations, while inducing-point and variational methods introduce further statistical approximations.

A stationary RBF kernel measures similarity through Euclidean distance in its input space. That distance may be appropriate in low dimensions but can be poorly aligned with the prediction task in high-dimensional inputs. For example, two face images can be far apart in pixel space while having similar orientations. A learned feature map can make Euclidean distance more task-relevant, so DKL learns a task-adapted kernel rather than the “right” kernel in any globally optimal sense. Figure 24.1 gives a stylised one-dimensional illustration of both the benefit and the caveat: feature warping changes the fit and the predictive uncertainty, and distance from the training data is measured in the learned feature space rather than the original input space.

The RBF kernel is a common base kernel, but it is not the only option. A spectral-mixture kernel [1763], for example, can encode quasi-periodic structure, leaving the neural feature map to capture complementary patterns.

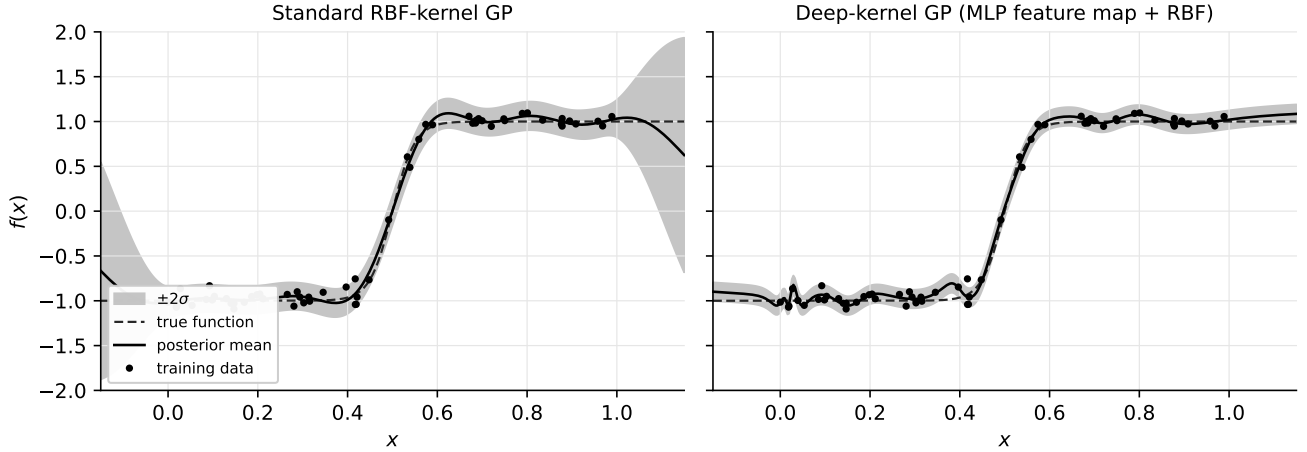


Figure 24.1: Illustrative deep-kernel regression for a target with a sharp transition, $f(x) = \tanh(20(x - 0.5))$, observed with independent Gaussian noise. The dashed line is the target, dots are the same 60 training observations in both panels, the solid line is the predictive mean, and the shaded region is the mean plus or minus two predictive standard deviations for a future noisy observation. Left: a GP with a stationary RBF kernel in the original input space. Right: a GP with the same RBF base kernel applied after a small multilayer-perceptron feature map, with all parameters learnt by marginal-likelihood maximisation. In this illustrative optimisation run, the deep kernel produces much narrower intervals, including outside the training range. The figure illustrates representation-dependent predictive behaviour, not superior calibration: narrower intervals are not automatically better calibrated and may reflect overconfidence under extrapolation.

24.3 Stochastic Variational Deep Kernel Learning

So far, we have considered DKL for Gaussian-likelihood regression, where the GP predictive distribution and marginal likelihood are available in closed form for fixed deep-kernel parameters. For classification and other non-Gaussian likelihoods, exact prediction and evaluation of the marginal likelihood are analytically intractable. Multi-class problems also require a construction for correlated outputs.

Wilson et al. [1766] address these challenges with *stochastic variational deep kernel learning* (SVDKL), building on sparse variational GP methods [737, 738, 1615]. SVDKL optimises a variational lower bound rather than the exact log marginal likelihood, supports mini-batch stochastic gradients, and accommodates non-Gaussian likelihoods and multiple outputs. For these reasons, it is also used for large-scale regression. Alternative formulations include Bradshaw et al. [202].

24.3.1 Details of SVDKL

We describe the construction of Wilson et al. [1766]; Section 24.5 discusses implementations that need not follow every detail of this formulation. Let

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N, \quad \mathbf{y}_i \in \{0, 1\}^C,$$

where C is the number of classes and $\mathbf{y}_i$ is a one-hot label. The neural feature extractor produces $\mathbf{z}_i = g_w(\mathbf{x}_i) \in \mathbb{R}^Q$. For each $j = 1, \dots, J$, let $\mathbf{z}_i^{(j)}$ denote the subset or projection of $\mathbf{z}_i$ supplied to latent GP f_j , and define

$$\mathbf{f}_i := (f_1(\mathbf{z}_i^{(1)}), \dots, f_J(\mathbf{z}_i^{(J)}))^T \in \mathbb{R}^J. \quad (24.7)$$

The collection of latent GPs forms an additive GP layer over subsets of the learnt features [e.g., 479]. Their outputs are mixed through $A \in \mathbb{R}^{C \times J}$, so the class logits are $A\mathbf{f}_i$. This mixing induces correlations between the outputs even when the latent GPs are independent. For a softmax observation model,

$$p(\mathbf{y}_i | \mathbf{f}_i, A) = \prod_{c=1}^C \left[\frac{\exp((A\mathbf{f}_i)_c)}{\sum_{c'=1}^C \exp((A\mathbf{f}_i)_{c'})} \right]^{y_{ic}}. \quad (24.8)$$

For GP j , let $H_j = \{\mathbf{z}_i^{(j)}\}_{i=1}^N$ denote its transformed training inputs, let Z_j contain m inducing inputs, let $\mathbf{f}_j$ collect its latent values on H_j , and let $\mathbf{u}_j$ collect its latent values on Z_j . The standard sparse-GP conditional [e.g.,

1354, 1615] is

$$\begin{aligned} p(\mathbf{f}_j | \mathbf{u}_j) &= \mathcal{N}(\mathbf{f}_j | K_{H_j Z_j}^{(j)} (K_{Z_j Z_j}^{(j)})^{-1} \mathbf{u}_j, \tilde{K}_j), \\ \tilde{K}_j &= K_{H_j H_j}^{(j)} - K_{H_j Z_j}^{(j)} (K_{Z_j Z_j}^{(j)})^{-1} K_{Z_j H_j}^{(j)}. \end{aligned} \quad (24.9)$$

Under the local kernel-interpolation approximation of Wilson and Nickisch [1764],

$$K_{H_j Z_j}^{(j)} \approx M_j K_{Z_j Z_j}^{(j)}, \quad K_{H_j H_j}^{(j)} \approx M_j K_{Z_j Z_j}^{(j)} M_j^\top, \quad (24.10)$$

where $M_j \in \mathbb{R}^{N \times m}$ is a sparse local-interpolation matrix. Substituting these approximations into Equation (24.9) sets the residual covariance to zero in the interpolated model and gives $\mathbf{f}_j \approx M_j \mathbf{u}_j$. This deterministic relation is an approximation introduced by interpolation, not an exact consequence of the original GP conditional.

Let $\mathbf{u} = (\mathbf{u}_1^\top, \dots, \mathbf{u}_J^\top)^\top$ and let $\mathbf{f}$ collect all latent GP values. SVDKL uses a factorised Gaussian variational distribution

$$q(\mathbf{u}) = \prod_{j=1}^J \mathcal{N}(\mathbf{u}_j | \boldsymbol{\mu}_j, \mathbf{S}_j). \quad (24.11)$$

Writing $\mathbf{y} = (\mathbf{y}_1^\top, \dots, \mathbf{y}_N^\top)^\top$, Jensen's inequality gives the evidence lower bound

$$\mathcal{L}(q; \gamma, A) := \mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y} | \mathbf{f}, A)] - \text{D}_{\text{KL}}(q(\mathbf{u}) \| p(\mathbf{u})) \leq \log p(\mathbf{y} | \gamma, A). \quad (24.12)$$

For fixed (γ, A) , maximising Equation (24.12) over q within the chosen variational family is equivalent to minimising the KL divergence from $q(\mathbf{u})$ to the exact posterior over the inducing variables. When (γ, A) are optimised jointly with the variational parameters, the same bound is used both for posterior approximation and for point estimation of the model parameters.

Because the likelihood factorises over observations, a uniformly sampled mini-batch $\mathcal{B} \subseteq \{1, \dots, N\}$ of size b gives the unbiased ELBO estimator

$$\hat{\mathcal{L}}_{\mathcal{B}} = \frac{N}{b} \sum_{i \in \mathcal{B}} \mathbb{E}_{q(\mathbf{f}_i)}[\log p(\mathbf{y}_i | \mathbf{f}_i, A)] - \text{D}_{\text{KL}}(q(\mathbf{u}) \| p(\mathbf{u})). \quad (24.13)$$

Here $q(\mathbf{f}_i)$ is the marginal induced by $q(\mathbf{u})p(\mathbf{f} | \mathbf{u})$. Stochastic gradients of this estimator update the variational parameters $\{(\boldsymbol{\mu}_j, \mathbf{S}_j)\}_{j=1}^J$ together with w , the base-kernel hyperparameters, and A . The expected log-likelihood is generally estimated by Monte Carlo for non-Gaussian observation models. Thus, SVDKL performs variational lower-bound optimisation, not exact marginal-likelihood training.

24.4 Kernel Estimation

Compared with more deeply nested probabilistic constructions such as deep Gaussian processes [361], standard DKL usually presents a simpler estimation problem. Nevertheless, practical performance depends strongly on how its many point-estimated parameters are trained.

Three strategies are common:

1. pre-train the neural network to learn w , freeze it, and fit only the base-kernel hyperparameters θ ;
2. use pre-training as a warm start and then fine-tune w and θ jointly;
3. learn $\gamma = (\theta, w)$ end-to-end from scratch.

The strategies differ in computational cost, susceptibility to overfitting, and reliance on a pre-training pipeline; none is uniformly best. In all three cases, standard DKL point-estimates w and θ , so its tractability should not be confused with a fully Bayesian treatment of uncertainty in the feature extractor.

A second choice is the training objective. For Gaussian-likelihood regression one can optimise the exact GP log marginal likelihood when dense computation is feasible, or an approximation to it when using scalable numerical methods. For non-Gaussian likelihoods and mini-batch training, SVDKL instead optimises the ELBO in Equation (24.12). The latter enables stochastic optimisation but introduces a variational approximation. Lotfi et al. [1081] propose the *conditional marginal likelihood*, which forms a posterior using one subset of the data and evaluates the marginal likelihood of the remainder; they report that this can mitigate marginal-likelihood underfitting in model selection.

Overfitting remains a legitimate concern. Ober et al. [1253] identify overfitting pathologies in marginal-likelihood-trained DKL and attribute them to excessive flexibility in the learned kernel. Wilson et al. [1771] dispute part of the accompanying explanation of the marginal-likelihood complexity term, arguing that it omits an additional data-fit dependence. These positions do not eliminate the practical need to check predictive performance and uncertainty on held-out data. A fully Bayesian treatment of the neural-network parameters can represent additional uncertainty but is substantially more expensive; standard regularisation such as weight decay is a more common practical safeguard.

When specialised base kernels such as the spectral-mixture kernel are used, tailored initialisation may be important [1763]. A sensible workflow is therefore to begin with a pre-trained and frozen feature extractor, fit the base kernel, and adopt joint fine-tuning, conditional marginal likelihood, or end-to-end training only when validation results justify the additional complexity.

24.5 DKL in Practice, Implementations, and Resources

Several mature frameworks support DKL. *GPyTorch* [580] integrates PyTorch neural-network modules with exact and approximate GP inference, GPU-accelerated linear algebra, and scalable variational methods. It is therefore well suited to prototyping DKL architectures and SVDKL variants, but the approximation and numerical assumptions of the selected GP method still apply.

A convenient starting point is the GPyTorch neural-network integration tutorial at https://docs.gpytorch.ai/en/stable/examples/06_PyTorch_MN_Integration_DKL/index.html. Before adding SKI or a variational approximation, it is often useful to test whether the default exact or iterative solvers are adequate for the dataset at hand. SVDKL provides a separate route to scale through inducing variables and mini-batch stochastic optimisation.

GPJax provides another deep-kernel tutorial at https://docs.jaxgaussianprocesses.com/_examples/deep_kernels/. For Bayesian optimisation, BoTorch [78] integrates with GPyTorch and supports DKL-based optimisation pipelines.

Chapter 25

Deep Gaussian processes

Bayesian deep learning strives to combine the expressive power of deep models with Bayesian uncertainty quantification. By placing distributions over the unknowns, we can propagate the uncertainty to predictions and embody Occam’s Razor. We saw in Chapter 23 that Gaussian process (GP) models also offer a Bayesian treatment over the unknowns and their predictions naturally incorporate both data fit and epistemic uncertainty. Unlike deep neural networks, GPs operate directly in *function space*. By specifying distributions over functions rather than over parameters they provide flexible, non-parametric, probabilistic approaches to function estimation in an analytically tractable manner [1382].

Traditional Gaussian process models have been extended to more expressive variants, for example by considering sophisticated covariance functions [473, 645] or by embedding GPs in more complex probabilistic structures [480, 1763] able to learn more powerful representations of the data. Going further, the research community has sought approaches which combine the advantages of GPs with the ability of deep learning architectures to achieve hierarchical feature learning, such as deep kernel learning which was discussed in the previous chapter.

In this chapter, we discuss an alternative approach to combining GPs with deep feature learning: Deep Gaussian processes (deep GPs). Deep GPs are deep architectures which compose functions drawn from GP priors. In particular, here we will consider constructions where the output of one layer is the input to the next as follows:

$$\mathbf{f}^{1:L}(\mathbf{x}) = \mathbf{f}^L(\mathbf{f}^{L-1}(\dots \mathbf{f}^1(\mathbf{x})) \dots), \quad (25.1)$$

where: $f^\ell \sim \mathcal{GP}(0, k_j^\ell(\cdot, \cdot))$.

Here, $\mathbf{f}$ denotes a vector-valued function and L is the total number of layers in the deep GP construction. Crucially, the overall process is *no longer Gaussian*, due to the nonlinear warping induced by each layer, and the model can learn much more complex interactions between data.

25.1 Composing functions and processes

25.1.1 Function composition for deep learning

Activation functions employed in neural networks are vector valued functions $\phi(\cdot)$ of an adjustable basis, which are controlled by a parameter matrix $\mathbf{U}$. In a similar way to generalised linear models, the final activation $\mathbf{g}(\mathbf{x})$ for a single layer neural network and input $\mathbf{x}$ is obtained by linearly weighting the bases using a matrix $\mathbf{V}$:

$$\mathbf{g}(\mathbf{x}) = \mathbf{V}^T \phi(\mathbf{U}\mathbf{x}). \quad (25.2)$$

Deep neural networks then take the form of a functional composition of the basis functions,

$$\mathbf{g}(\mathbf{x}) = (\mathbf{V}^L)^T \phi^L(\mathbf{U}^L \mathbf{g}^{L-1}(\dots \mathbf{U}^2 \mathbf{g}^1(\mathbf{x}))), \quad (25.3)$$

where L denotes the total number of layers and we represent the overall composition with the unindexed notation $\mathbf{g}(\cdot)$. We can alternatively re-write the above expression by using equation (25.2) to replace all $\mathbf{g}^\ell(\cdot)$ terms:

$$\mathbf{g}(\mathbf{x}) = (\mathbf{V}^L)^T \phi^L(\mathbf{W}^L \phi^{L-1}(\dots \mathbf{W}^2 \phi^1(\mathbf{U}^1 \mathbf{x}))), \quad (25.4)$$

where $\mathbf{W}^\ell$ is the matrix mapping between each set of basis functions and is typically treated as a parameter to be learned. A serious challenge for deep networks, when trained in a feed-forward manner, is overfitting. As the

number of layers and the number of basis functions per layer increases, a very powerful but also highly parametrised representation is formed. The matrix $\mathbf{W}^\ell$ has size $d^\ell \times d^{\ell-1}$, where d^ℓ is the number of basis functions in the ℓ th layer.

25.1.2 Weight matrix factorisation

One approach to dealing with the many parameters contained in the matrices $\mathbf{W}^\ell$, is to replace them with a lower rank form. This draws inspiration from the first function composition formulation shown in equation (25.3), but now the low rank decomposition is made explicit and clear. We select $r^\ell < d^\ell, d^{\ell-1}$ so as to obtain:

$$\mathbf{W}^\ell = \mathbf{U}^\ell (\mathbf{V}^{\ell-1})^\top,$$

where $\mathbf{U}^\ell \in \mathbb{R}^{d^\ell \times r^\ell}$ and $\mathbf{V}^{\ell-1} \in \mathbb{R}^{d^{\ell-1} \times r^\ell}$.

Denil et al. [411] have empirically shown that trained neural networks might be well approximated by low rank matrices as evidenced by the ability to predict one part of the weight matrix given another. The approach of “dropout” [1552] is also widely applied to control the complexity of the model implying the models are over parameterised. Substituting the low rank form into the compositional structure for the deep network we have

$$\mathbf{g}(\mathbf{x}) = (\mathbf{V}^L)^\top \phi^L(\mathbf{U}^L (\mathbf{V}^{L-1})^\top \phi^{L-1}(\dots \mathbf{U}^2 (\mathbf{V}^1)^\top \phi^1(\mathbf{U}^1 \mathbf{x}))).$$

We can now identify the following form inside the functional decomposition,

$$\mathbf{g}^\ell(\mathbf{z}) = \mathbf{V}^\ell \phi^\ell(\mathbf{U}^\ell \mathbf{z}) \quad (25.5)$$

and once again obtain a functional composition,

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}^L(\mathbf{f}^{L-1}(\dots \mathbf{f}^1(\mathbf{x}))). \quad (25.6)$$

The standard deep network is recovered when we set $r^\ell = \min(d^\ell, d^{\ell-1})$.

25.1.3 Process composition with latent variables

To obtain a deep Gaussian process we consider a nested structure as shown above with equation (25.6). The deep Gaussian process is recovered by keeping r^ℓ finite and allowing $d^\ell \rightarrow \infty$ for all layers. Of course, the mappings in the Gaussian process are treated probabilistically and integrated out¹, rather than optimised, so despite the increase in layer size the parameters in the resulting model are many times fewer than those in a standard deep neural network. Further, Duvenaud et al. [481] additionally analyse the effect of dropout in the GP limit, showing it produces spike-and-slab lengthscale priors.

The latent variable construction above is not the only route to a deep GP. Alternative formulations include viewing the composition as a stochastic differential equation whose layers become continuous [728], representing the deep GP as a state-space model amenable to sequential filtering [1867], or describing it through composed covariance operators [463]. We discuss these perspectives further below.

The compositional idea also places deep GPs in a broader landscape of models. Warped GPs [1529] constitute a single-layer precursor, in which the output of a GP is transformed through a parametric monotonic function (which can also admit a GP prior [363, 986]), while normalising flows [1290, 1390] likewise construct flexible distributions by composing nonlinear transformations, although through deterministic, invertible parametric maps rather than draws from nonparametric function priors.

To generalise a Gaussian process to the deep domain, rather than assuming that a data observation, $\mathbf{y}$, is a draw from a noise corrupted Gaussian process prior, we instead make use of a functional composition,

$$\begin{aligned} \mathbf{y} &= \mathbf{f}^{1:L} + \boldsymbol{\epsilon} \\ &= \mathbf{f}^L(\mathbf{f}^{L-1}(\dots \mathbf{f}^1(\mathbf{x}))) + \boldsymbol{\epsilon}. \end{aligned} \quad (25.7)$$

This is the same form as for standard neural networks, shown in equation (25.6), but now each function in the composition, $\mathbf{f}^\ell(\cdot)$, is a draw from a Gaussian process. However, the overall prior $\mathbf{f}^{1:L}$ is no longer a Gaussian process. In the above equation, $\boldsymbol{\epsilon}$ represents the noise, and we can introduce Gaussian noise in every layer. In this

¹If the mappings are instead fixed, then we obtain a standard (shallow) GP with a hierarchical covariance function, as noted and investigated by Duvenaud et al. [481].

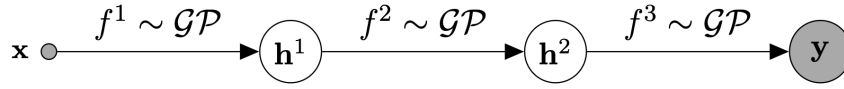


Figure 25.1: A deep Gaussian process with two hidden layers.

case, we can collect the noise corrupted function instantiations (corresponding to n data points) into a vector for each layer, so that we get the following recursive definition:

$$\mathbf{h}^\ell = \mathbf{f}^\ell(\mathbf{h}^{\ell-1}) + \boldsymbol{\epsilon}^\ell. \quad (25.8)$$

From this recursive definition, we obtain the whole hierarchical construction if we denote $\mathbf{x} \triangleq \mathbf{h}^0$ and consider $\mathbf{y}$ as the last layer. The corresponding graphical model for a deep Gaussian process is illustrated in Figure 25.1.

As can be seen, the generative procedure defined by the function $\mathbf{f}^{1:L}$ can be represented in a graphical model as a deep architecture where each layer corresponds to a generative procedure associated with a function $\mathbf{f}^\ell$.

Given equation (25.7), it is clear that this chapter develops a full probabilistic model through *process composition*. Process composition has the appealing property of retaining the theoretical qualities of the underlying stochastic process (such as Kolmogorov consistency) whilst providing a richer class of process priors. For example, for deep Gaussian processes, it has been shown that for particular assumptions of covariance function parameters, the derivatives of functions sampled from the process have a marginal distribution that is heavier tailed than a Gaussian [481]. In contrast, it is known that in a standard Gaussian process the derivatives of functions sampled from the process are jointly Gaussian with the original function. In the next section we compare the properties of deep Gaussian processes with the properties of traditional deep learning methods.

25.1.4 Inference challenges and expressiveness

We have so far focused on a deep GP construction with a hierarchy of intermediate latent variables. To highlight the inference challenges and expressiveness of the model, it is instructive to compare this construction to the generative deep learning approach that was dominant when deep GPs were first introduced: restricted Boltzmann machines (RBMs), stacked to construct deep belief networks. Deep belief networks have since been superseded as the workhorses of deep generative modelling, first by variational autoencoders [892, 1391] and more recently by diffusion models (Chapter 50), but the comparison remains instructive: like the deep GP, a deep belief network is a hierarchy of latent variables whose training requires marginalising the intermediate layers, and the contrast cleanly exposes what changes when each layer is a nonparametric function rather than a parametric mapping.

Various approximate inference techniques (such as contrastive divergence) are used for estimating model parameters in RBMs. A significant amount of work has then to be done with annealed importance sampling if even the *likelihood*² of a data set under the RBM model is to be estimated [1450]. When deeper hierarchies are considered, the estimate is only of a lower bound on the data likelihood. By considering sigmoidal activation functions, $\sigma(z) = (1 + \exp(-z))^{-1}$, we obtain the conditional probability of a single hidden unit in a deep belief network given its parents as

$$p(y|\mathbf{x}) = \sigma(\mathbf{U}\mathbf{x})^y (1 - \sigma(\mathbf{U}\mathbf{x}))^{(1-y)}.$$

The conditional density of the output y depends only on a linear weighted sum of the inputs $\mathbf{x}$. The representational power of a Gaussian process in the same role is significantly greater than that of an RBM. For the GP the corresponding likelihood is over a continuous variable, but it is a nonlinear function of the inputs,

$$p(y|\mathbf{x}) = \mathcal{N}(y|f(\mathbf{x}), \beta^{-1}).$$

In this case the likelihood is dependent on a mapping function, $f(\cdot)$, rather than a set of intermediate parameters, $\mathbf{U}$. By placing a GP prior over f , we can analytically integrate out the mappings. In the RBM, instead, the model likelihood is estimated and maximised with respect to the parameters, $\mathbf{U}$. For the RBM, marginalising $\mathbf{U}$ is not analytically tractable. Note that the two approaches can be mixed if

$$p(y|\mathbf{x}) = \sigma(f(\mathbf{x}))^y (1 - \sigma(f(\mathbf{x})))^{(1-y)},$$

²Emphasis is used to clarify we are referring to the model likelihood, not the marginal likelihood required in Bayesian model selection.

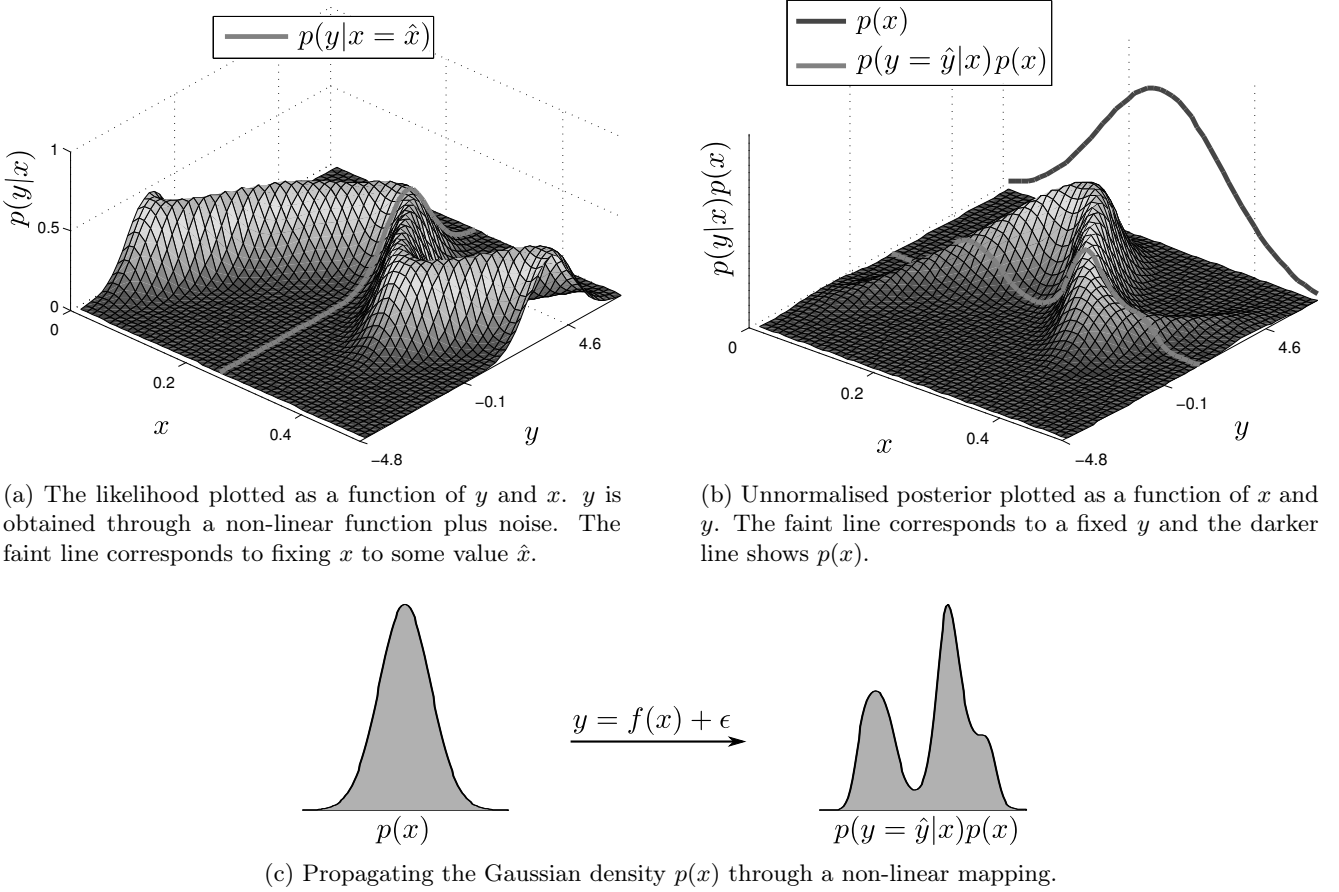


Figure 25.2: A Gaussian distribution propagated through a non-linear mapping. Figures a and b show the likelihood and unnormalised posterior respectively. The posterior density (plotted for given data using a faint line in figure b and on the right in figure c) is multimodal and difficult to normalise.

which recovers a GP classification model. Analytic integration is no longer possible though, and a common approach to approximate inference is the expectation propagation algorithm [see e.g. 1382]. Expectation propagation has also been applied to deep GPs [217]; we discuss this and other inference approaches in Section 25.4. Here we focus on the variational approximation.

Inference in deep models requires marginalisation of the stacked input spaces, denoted as $\{\mathbf{h}^\ell\}_{\ell=1}^L$ in equation (25.8). These *intermediate* spaces are typically treated as *latent* variables³, which in the case of the RBM are binary variables. The number of the terms in the sum scales exponentially with the input dimension rendering it intractable for anything but the smallest models. In practice, sampling and, in particular, the contrastive divergence algorithm, are used for training. Similarly, marginalising the intermediate, latent spaces in the deep GP is analytically intractable: despite the simplicity of the Gaussian density its propagation through non-linear mappings is challenging. This is illustrated in Figure 25.2.

For this reason, the hierarchical GP-LVM [985], which constituted the first approach to stacking GPs, was defined within a maximum a posteriori (MAP) framework where the latent variables were maximised rather than marginalised out. For this MAP approach to work, however, a strong prior is required on the top level of the hierarchy. Further, MAP learning prohibits model selection because no estimate of the marginal likelihood is available.

Most deep algorithms require a large amount of data to perform learning. Fitting such models to smaller data sets and using Bayesian approaches to deal with the complexity seems completely futile when faced with the associated computational intractabilities. However, we know that humans are able to perform inductive reasoning (equivalent to concept generalisation) with only a few examples [1594]. This provokes the question as to whether deep structures and the learning of abstract structure can be undertaken in *smaller* data sets. For smaller data

³Some of the variables can also be treated as observed, e.g. in the upper most layer of the hierarchy where we might include the data label or inputs for regression.

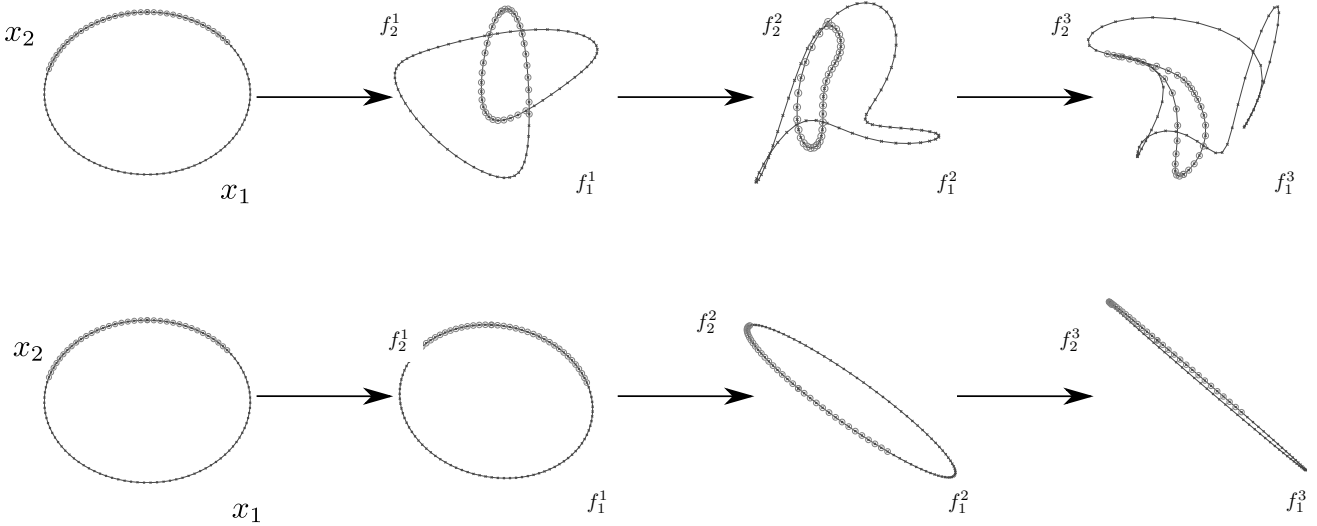


Figure 25.3: Samples from a deep GP showing the generation of features. The upper plot shows the successive non-linear warping of a two-dimensional input space. The circles correspond to specific locations in the input space for which a feature (a “loop”) is created in layer 1. As can be seen, as we traverse the hierarchy towards the right, this feature is maintained in the next layers and is potentially further transformed. The bottom plot is similar as the upper plot, but a linear covariance function is used (equivalent to performing stacked principal component analysis (PCA) for training given observed data). As can be seen, linear transformations seem nonsensical in this setting, since multiplying an input by two matrices (two linear transformations) is equivalent to multiplying the input with a single combined matrix (one linear transformation).

sets, questions of generalisation arise: to demonstrate such structures are justified it is useful to have an objective measure of the model’s applicability. This chapter demonstrates the ability of deep Gaussian processes to learn even from small training sets. This property can be attributed to their fully Bayesian nature.

25.1.5 Characteristics of deep GPs as priors

Deep GPs compose nonlinear mappings to yield much richer prior function classes compared to single layer GPs. To start with, a single-layer GP defines a distribution over functions which is Gaussian: any finite collection of function values has a joint multivariate Gaussian distribution. This Gaussianity imposes certain limitations on expressiveness. For instance, function derivatives in a GP are also Gaussian distributed which means that very sharp changes or large jumps in the function space are a priori unlikely under a GP. On the other hand, the output of a deep GP is not Gaussian distributed in general; even if each layer is a GP, the marginal distribution $p(y|x)$ as modelled by $y = \mathcal{DGP}(x)$, where $\mathcal{DGP}$ denotes the composite deep GP mapping, is an integral of Gaussian distributions over random inputs, which can produce heavy tails, skewness or multimodality. More generally, as the number of layers increases, the distribution of outputs given an input can develop increasingly non-trivial shapes, as discussed by Duvenaud et al. [481] who also elaborate on caveats related to this characteristic.

Furthermore, as a function prior, a deep GP can induce non-stationarity and input warping. By composing multiple GPs, the effective covariance function of the input-output mapping becomes input-dependent. For instance, a single GP with a stationary kernel (such as the exponentiated quadratic) is limited in that it has a single global lengthscale of correlation. A deep GP can *learn* location-varying lengthscales: the first GP layer might warp the input x into a new space h^1 , stretching or contracting regions differently. The second layer then operates on h^1 and so on. The result is that the final function can exhibit non-stationary behaviour, even if each layer used stationary kernels. For instance, one part of the input space might effectively be magnified (leading to fine-scale, rapidly varying output), while another part is shrunk (output varies more slowly), because the latent h^1 learned that certain inputs require more detail. This is illustrated in Figures 25.3, 25.4 and 25.7.

Another view is to consider what covariance function a deep GP implies for the direct input-output relationship. In a few cases, for example simple linear compositions or analogies to infinite-width neural networks, one can derive an equivalent covariance function for the composed function analytically. Classic results by Neal [1227] and Williams [1759] showed that a single-layer Bayesian neural network with an infinite number of hidden units converges to a GP with a closed-form covariance function for certain activation functions. However, for a general

deep GP with nonlinear covariance at each layer, there is generally no closed-form kernel for the overall composed mapping.

An important line of work has extended the single-layer GP limit to deep networks. Dutordoir et al. [477] showed that a fully-connected Bayesian neural network of fixed depth, with i.i.d. weight priors converges to a single GP as layer widths are taken to infinity sequentially. The limiting kernel $\mathbf{K}^\ell$ is obtained by a deterministic recursion: each layer applies a map to the previous layer’s kernel, determined by the activation function and the weight and bias variances. de G. Matthews et al. [386] made this result rigorous for simultaneous width growth across all layers, resolving the key technical obstacle that the standard central limit theorem no longer applies in deep architectures (since the required independence assumptions fail) by reformulating the layerwise sums as exchangeable sequences and invoking a central limit theorem for exchangeable arrays. Crucially, in both cases the limit is a single GP, not a deep GP: depth enters only through the kernel recursion, and the hierarchy collapses into a fixed, deterministic kernel. This limiting object is sometimes called the neural network GP (NNGP) and is best understood as a *deep kernel GP* rather than a deep GP in the sense of Damianou and Lawrence [361]. The relationship to deep kernel learning, where a neural network is used to parameterise the kernel of a (single-layer) GP, is discussed in Chapter 24.

Agrawal et al. [9] resolve the apparent disconnect between these GP limits and deep GPs. They unify the NNGP and deep GP perspectives by showing that when some hidden layers are kept at finite width (bottlenecks) while others tend to infinity, the network converges to a composition of Gaussian processes, i.e. a deep Gaussian process. These bottleneck layers introduce shared latent representations that induce dependence between outputs and preserve discriminative structure through depth. In contrast, when all layers are taken to infinite width, the model reduces to a single Gaussian process. Conversely, Pleiss and Cunningham [1330] demonstrate that width and depth have opposite effects in hierarchical models: while depth accentuates non-Gaussianity, increasing the output dimensionality of the intermediate layers causes a deep GP to converge toward a standard GP. Because deep GPs are already nonparametric, this collapse cannot be attributed to gaining capacity; it is a structural consequence of width itself, reinforcing the importance of narrow bottleneck layers for preserving hierarchical structure. Since Bayesian neural networks constitute a subclass of deep GPs with degenerate, finite-capacity layers, these structural findings extend to standard neural network architectures as well.

Connecting in the reverse direction, Dutordoir et al. [477] construct deep sparse GP models whose layerwise mean propagation matches the forward pass of a standard neural network through interdomain inducing-feature constructions. This provides a practical bridge: neural networks can serve as point estimates for deep GP inference, and a trained neural network can be used to initialise a deep GP.

The covariance-function perspective connects to a broader class of constructions of deep Gaussian processes, including formulations based on covariance operators and convolution [463]. Dunlop et al. [463] place these approaches within a unified framework in which the sequence of functions across layers forms a Markov chain, and analyse its ergodicity properties to characterise an effective depth: in ergodic settings, the process admits a stationary distribution and additional layers yield little further benefit beyond the mixing time, whereas in non-ergodic settings the behaviour can remain non-trivial or unstable as depth increases. From a statistical perspective, Finocchio and Schmidt-Hieber [519] complement this view by showing that deep Gaussian process priors can achieve (near-)minimax posterior contraction rates for regression functions with compositional structure, demonstrating that depth can be advantageous in such settings.

A practical concern related to depth is the pathology identified by Duvenaud et al. [481], whereby the prior’s expressive degrees of freedom can collapse as the number of layers grows. In the context of variational inference, this can manifest as layer collapse: mean-field approximations may lead to intermediate layers being effectively ‘turned off’, with their contribution diminishing as the signal-to-noise ratio vanishes [1453]. This pathology is not intrinsic, however: as we discuss in the following section, architectural modifications at the bottleneck layers can alleviate it.

25.1.6 Modelling bottleneck layers

The preceding section established that bottleneck layers (narrow intermediate layers) are the load-bearing structural element of a deep GP. We now turn to the complementary question how these bottleneck layers can be modelled. We first discuss architectural choices aimed at preserving the contribution of intermediate layers, directly addressing the pathology just mentioned, and then turn to a deeper modelling axis: whether the bottleneck is represented as an explicit latent variable in the generative model, or left implicit in the composition.

To mitigate the pathology noted by Duvenaud et al. [481] and discussed in the previous section, the authors propose architectures that connect the input directly to each layer, and Salimbeni and Deisenroth [1453] introduce a closely related modification in the deep GP setting by equipping each layer with a linear mean function $m(\mathbf{x}) = \mathbf{W}\mathbf{x}$.

This acts as a residual connection: even if a layer’s stochastic component contributes little, the input can still propagate through the linear mean. This modification has been shown to alleviate the degeneracies identified by Duvenaud et al. [481] and is consistent with the analysis of deep GP priors in Salimbeni and Deisenroth [1453]. More recently, de Souza et al. [388] propose *thin and deep* architectures that combine low-dimensional latent representations with interpretable kernel parametrisations at each layer, addressing these pathologies through architectural design while preserving the benefits of depth.

Beyond architectural modifications, the bottleneck layer itself admits two fundamentally different modelling treatments, distinguished by whether it is represented as an explicit latent variable in the generative model or only implicitly via the composition. In the explicit-latent view, followed by Damianou and Lawrence [361], each bottleneck is a random variable $\mathbf{h}^\ell = f^\ell(\mathbf{h}^{\ell-1}) + \epsilon^\ell$, distinct from the GP function values, with Gaussian noise ϵ^ℓ injected between every pair of layers. Each layer then has the structural form of a GP-LVM, and inference includes estimating $p(\{\mathbf{h}^\ell\}_{\ell=1}^L | \text{Data})$, the posterior over the intermediate bottlenecks. In the implicit-latent view, followed by Salimbeni and Deisenroth [1453], the inter-layer noise is absorbed into the covariance function and there is no separately named bottleneck variable: the state at layer ℓ is the GP function value $\mathbf{f}^\ell$ itself, and the key interest is the induced marginal process $y(x) = \mathcal{DGP}(x)$, with intermediate $\mathbf{f}^\ell$ regarded as instrumental rather than as objects of inference.

This distinction looks small, as it just amounts to whether noise is injected between layers, but its consequences are substantial. The explicit-latent view makes each layer a GP-LVM and so it places the model squarely in the lineage of representation learning: the inferred intermediate bottlenecks are analogues of the embeddings obtained from intermediate layers of deep networks, and the generative interpretation readily accommodates supervised, unsupervised, and semi-supervised [362] learning, as well as the modelling of high-dimensional data. The implicit-latent view instead emphasises the induced function $y(x) = \mathcal{DGP}(x)$ and its properties rather than the learned inner representations, making function learning the natural framing; this reorientation has also motivated different, more scalable inference schemes for the supervised learning scenario. Finally, the implicit-latent view is sometimes taken to include nested GP models, where a GP prior is placed on the parameters of a GP as in Zhao et al. [1867].

25.1.7 Outline for the rest of this chapter

After establishing these alternative modelling choices for deep GP bottleneck layers, we proceed in this chapter by diving deeper into two families of approaches which are particularly instructive. Firstly, we discuss the explicit-latent view; this has been the initial introduction of deep GPs with variational Bayesian inference by Damianou and Lawrence [361] and constitutes an intuitive description of the generative model and its relation to BDL. This view also has been amenable to a variety of inference schemes, including variational inference which has been discussed in this book. Secondly, we discuss approaches such as Salimbeni and Deisenroth [1453] where estimation of the distribution of intermediate latents is not the key focus, further motivating a relaxation for the requirement that inner expectations be evaluated analytically. In turn, this gives rise to a Monte-Carlo approximation of the predictive (approximate) posterior which unlocks both scalability and stability.

Overall, by elaborating on the above two approaches, their different points of focus and how their associated variational inference schemes deeply relate to each other, we hope that the reader will get a rounded understanding of deep GPs, also facilitated by the previous discussion of different deep GP constructions.

We will give a detailed explanation of the VI-based inference schemes used in these approaches to expose the readers to the precise approximations made; this is important since the central technical challenge of deep GPs is inference. Indeed, we have established deep GPs are BDL models operating in function space directly, giving us the combined benefit of: (a) function space inference (b) fully Bayesian treatment and (c) non-linear mappings; however exactly this induced requirement of propagating densities through non-linearities means that we can no longer apply the analytically tractable methodologies of shallow GPs, nor can we employ directly deep learning optimisation techniques which work in the parameter space (they are not required to propagate densities). The intra-layer as well as inter-layer intractabilities mean that standard variational inference also fails. Consequently, practical trade-offs and challenges for deep GP constructions and usage depend critically on inference. We will complement the thorough discussion of the VI schemes by additionally discussing other inference schemes besides variational inference. Finally, we will close this chapter with an illustrative example.

25.2 Deep Gaussian processes with latent variables

This section describes the deep GP framework in more detail, starting with the probabilistic definition and then developing the variational inference methodology used for training.

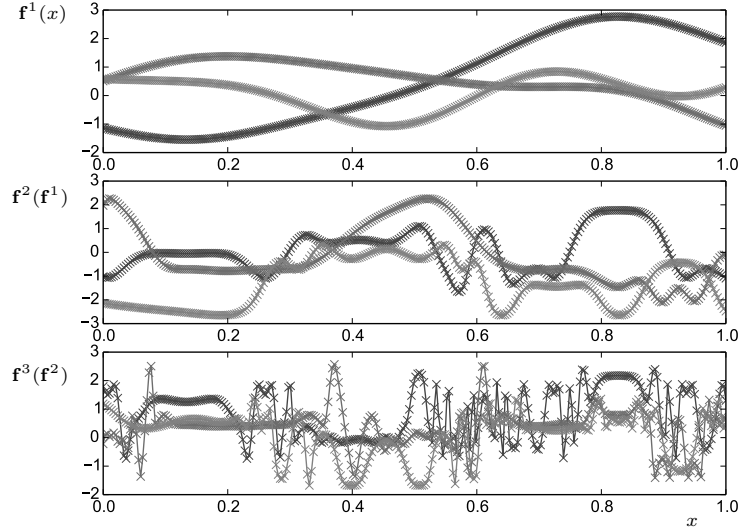


Figure 25.4: Samples from a deep Gaussian process with three-dimensional hidden and output nodes. A linear input x (not depicted) was propagated through two hidden layers (top and middle plot) to the output layer (bottom plot).

25.2.1 Probabilistic definition

The deep Gaussian process class of probabilistic models is now formally described. To do so, single-dimensional variables will again be considered, although it is underlined that this is only for simplicity and without loss of generality for models with conditionally independent output dimensions. Further, the analysis is made by firstly considering the unsupervised learning scenario, where a hierarchy of latent spaces is assumed to generate observed outputs.

The deep GP consists of a cascade of L hidden layers of latent variables, $\{\mathbf{h}^\ell\}_{\ell=1}^L$ with the observed outputs being placed in the leaves of the hierarchy. An intermediate node $\mathbf{h}^\ell \in \mathbb{R}^n$ associated with layer ℓ constitutes the output of that layer and acts as an input for the generative procedure of the subsequent layer, $\ell + 1$. Gaussian processes govern the mappings between the layers. Each mapping corresponds to a separate GP, having covariance function k^ℓ and hyperparameters θ^ℓ . Further, every set of latent variables $\mathbf{h}^\ell$ is obtained from the functional output of the previous layer with the addition of Gaussian noise $\epsilon^\ell \sim \mathcal{N}(0, (\beta^\ell)^{-1}\mathbf{I})$. Therefore, a single layer of the deep GP is effectively a GP-LVM, just as a single layer of a deep belief network is an RBM. The joint distribution of a deep GP model with L hidden layers can be written as:

$$p(\mathbf{y}, \{\mathbf{h}^\ell\}_{\ell=1}^L) = p(\mathbf{y}|\mathbf{h}^L)p(\mathbf{h}^L|\mathbf{h}^{L-1}) \cdots p(\mathbf{h}^2|\mathbf{h}^1)p(\mathbf{h}^1), \quad (25.9)$$

where $p(\mathbf{h}^1) = \mathcal{N}(\mathbf{h}^1|\mathbf{0}, \mathbf{I})$. As for the conditional probabilities, they can be expanded as:

$$p(\mathbf{h}^\ell|\mathbf{h}^{\ell-1}) = \int p(\mathbf{h}^\ell|\mathbf{f}^\ell)p(\mathbf{f}^\ell|\mathbf{h}^{\ell-1})d\mathbf{f}^\ell, \quad (25.10)$$

where inside the integral we have the likelihood $p(\mathbf{h}^\ell|\mathbf{f}^\ell) \sim \mathcal{N}(\mathbf{f}^\ell, (\beta^\ell)^{-1}\mathbf{I})$ and the conditional $p(\mathbf{f}^\ell|\mathbf{h}^{\ell-1}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{f^{\ell-1}f^{\ell-1}})$. Notice that the covariance matrix for each layer is evaluated with the corresponding covariance function evaluated at the inputs of the previous layer, that is, $\mathbf{K}_{f^{\ell-1}f^{\ell-1}} = k^\ell(\mathbf{h}^{\ell-1}, \mathbf{h}^{\ell-1})$. The observations are included in this recursive formulation by considering the special notation $\mathbf{h}^{L+1} \triangleq \mathbf{y}$.

Figure 25.4 shows examples of samples drawn from the deep GP architecture, where the top layer was chosen deterministically. This figure makes evident that deep GPs constitute much more powerful priors compared to traditional GPs (observe, for example, the long range correlations in the samples). However, although sampling from a deep GP is easy, training it is challenging. MAP optimisation for $\{\mathbf{h}^\ell\}_{\ell=1}^L$ turns all hidden nodes into extra model parameters, creating a severe regularisation problem [985]. Here we instead focus on marginalising out all hidden layers in a Bayesian paradigm, which drastically reduces the number of parameters (each layer mainly contributes variational rather than model parameters) and enables automatic structure discovery through the model evidence and automatic relevance determination (ARD) [364].

However, the nonlinearities introduced by the GP covariance functions make the Bayesian treatment of this model challenging. To demonstrate this, consider a deep GP with two hidden layers. Then, the marginal likelihood

can be written as:

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{h}^2) \left(\int p(\mathbf{h}^2|\mathbf{h}^1) p(\mathbf{h}^1) d\mathbf{h}^1 \right) d\mathbf{h}^2. \quad (25.11)$$

The nested integral can be written according to equation (25.10) as:

$$p(\mathbf{h}^2) = \int p(\mathbf{h}^2|\mathbf{f}^2) p(\mathbf{f}^2|\mathbf{h}^1) p(\mathbf{h}^1) d\mathbf{h}^1 d\mathbf{f}^2 = \langle p(\mathbf{h}^2|\mathbf{h}^1) \rangle_{p(\mathbf{h}^1)} \quad (25.12)$$

where $\langle \cdot \rangle_p$ denotes expectation with respect to p ; the above integral is intractable, because the density $p(\mathbf{h}^1)$ cannot be propagated through the function $p(\mathbf{f}^2|\mathbf{h}^1)$ which is constructed by a nonlinear covariance function $k^2(\mathbf{h}^1, \mathbf{h}^1)$. A similar situation arises in every layer when deeper architectures are employed. That is, if there was an additional layer we would have to compute the expectation $\langle p(\mathbf{h}^3|\mathbf{h}^2) \rangle_{\tilde{p}(\mathbf{h}^2)}$, where $\tilde{p}(\mathbf{h}^2) = \langle p(\mathbf{h}^2|\mathbf{h}^1) \rangle_{p(\mathbf{h}^1)}$ is coming from equation (25.12). We can, thus, formalise the problem by writing the recursive definition

$$\tilde{p}(\mathbf{h}^\ell) = \langle p(\mathbf{h}^\ell|\mathbf{h}^{\ell-1}) \rangle_{\tilde{p}(\mathbf{h}^{\ell-1})}$$

of the densities required for the calculation of the marginal likelihood and observing that all required expectations are intractable, as demonstrated for the two hidden layer architecture above. In the next section we develop variational methodologies which solve this intractability.

25.2.2 Variational inference

Inducing variables and sparse GP conditionals. Here we will demonstrate the variational framework employed by Damianou and Lawrence [361] to introduce tractable deep GPs. We also point the reader's attention to Chapter 12 and the chapters that follow it for more information on variational inference. First, following the variational sparse GP framework [1615], we augment the probability space of every layer with m inducing function values $\mathbf{u}^\ell$ evaluated at pseudo-inputs $\mathbf{x}_u^\ell$ [1354] (Figure 25.5), treated in depth in Chapter 38. The inducing inputs will be omitted from our expressions, since they always appear in the conditioning set of any probability involving $\mathbf{u}$. For layers with latent inputs ($\ell \geq 2$ in the unsupervised formulation), the inducing inputs $\mathbf{x}_u^\ell$ must be placed in the same latent space as $\mathbf{h}^{\ell-1}$, whose geometry is itself being learned; both are optimised jointly, and poor coordination between them can degrade the sparse approximation.

Given these inducing variables, the standard sparse GP decomposition [1354] gives the conditional of the latent function values at layer ℓ :

$$p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{h}^{\ell-1}) = \mathcal{N}(\mathbf{f}^\ell|\mathbf{a}^\ell, \tilde{\mathbf{K}}^\ell), \quad (25.13)$$

where:

$$\begin{aligned} \mathbf{a}^\ell &= \mathbf{K}_{f^{\ell-1}u^{\ell-1}} \mathbf{K}_{u^{\ell-1}u^{\ell-1}}^{-1} \mathbf{u}^\ell \\ \tilde{\mathbf{K}}^\ell &= \mathbf{K}_{f^{\ell-1}f^{\ell-1}} - \mathbf{K}_{f^{\ell-1}u^{\ell-1}} \mathbf{K}_{u^{\ell-1}u^{\ell-1}}^{-1} \mathbf{K}_{u^{\ell-1}f^{\ell-1}}. \end{aligned} \quad (25.14)$$

Variational compression within each layer. Even with the inducing point expansion, we cannot propagate a density of $\mathbf{h}^{\ell-1}$ through the nonlinear $p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{h}^{\ell-1})$. Following Titsias [1615], Titsias and Lawrence [1616], we set $q(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{h}^{\ell-1}) = p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{h}^{\ell-1})$, which yields a tractable bound; this bound becomes tight in the limit where u^ℓ is a sufficient statistic for f^ℓ [360, 361]. Applying Jensen's inequality within each layer then yields a per-layer contribution to the bound:

$$\mathcal{L}^\ell = \log \mathcal{N}(\mathbf{h}^\ell|\mathbf{a}^\ell, (\beta^\ell)^{-1}\mathbf{I}) - \frac{\beta^\ell}{2} \text{tr} \tilde{\mathbf{K}}^\ell. \quad (25.15)$$

Equation (25.15) lower bounds $\log p(\mathbf{h}^\ell|\mathbf{u}^\ell, \mathbf{h}^{\ell-1})$: the Gaussian uses only the noise covariance $(\beta^\ell)^{-1}\mathbf{I}$ in place of the full $\tilde{\mathbf{K}}^\ell + (\beta^\ell)^{-1}\mathbf{I}$, while the trace term penalises the residual GP variance not captured by the inducing variables [1615]. Conditioned on $\mathbf{u}$, this per-layer bound factorises across data points.

Integrating out hidden variables and inducing outputs. It remains to marginalise out the hidden variables $\{\mathbf{h}^\ell\}$ across layers and the inducing function values $\{\mathbf{u}^\ell\}$. To do so, we introduce a variational distribution

$$\mathcal{Q} = \prod_{\ell=1}^L q(\mathbf{u}^{\ell+1}) q(\mathbf{h}^\ell). \quad (25.16)$$

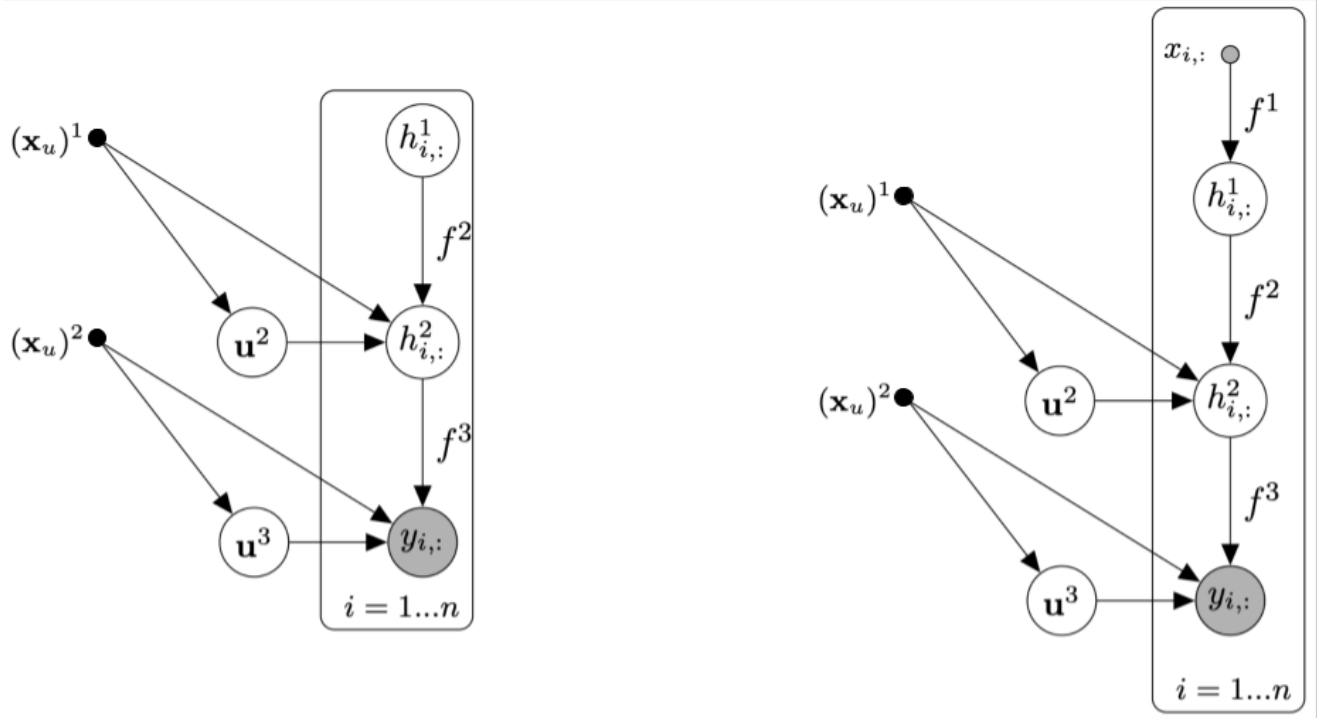


Figure 25.5: A graphical model representation of: the unsupervised (left) and the supervised (right) deep GP structure with inducing variables and $L = 2$ hidden layers. Nodes arranged horizontally form part of the same layer.

where both $q(\mathbf{h}^\ell)$ and $q(\mathbf{u}^\ell)$ are chosen to be Gaussian, with $q(\mathbf{h}^\ell)$ factorised across data points (diagonal covariance per data point) and $q(\mathbf{u}^\ell)$ having full covariance. Notice that factorising $q(\{\mathbf{h}^\ell\})$ across layers is a (sub-optimal) mean-field approximation.

Applying Jensen's inequality with this $\mathcal{Q}$ and substituting the per-layer bounds from equation (25.15), we obtain the variational lower bound $\mathcal{F} \leq \log p(\mathbf{y})$ (see Damianou [360], Damianou and Lawrence [361] for the full derivation):

$$\begin{aligned} \mathcal{F}(q\{\mathbf{H}^\ell\}, q\{\mathbf{U}^\ell\}) &= \sum_{\ell=2}^{L+1} \left(\langle \log \mathcal{N}(\mathbf{h}^\ell | \mathbf{a}^\ell, (\beta^\ell)^{-1} \mathbf{I}) \rangle_{q(\mathbf{h}^{\ell-1})q(\mathbf{h}^\ell)q(\mathbf{u}^\ell)} - \frac{\beta^\ell}{2} \langle \text{tr} \tilde{\mathbf{K}}^\ell \rangle_{q(\mathbf{h}^{\ell-1})} \right) \\ &\quad - \text{D}_{\text{KL}}(q(\mathbf{h}^1) \| p(\mathbf{h}^1)) - \sum_{\ell=2}^{L+1} \text{D}_{\text{KL}}(q(\mathbf{u}^\ell) \| p(\mathbf{u}^\ell)) + \sum_{\ell=2}^L \mathcal{H}q(\mathbf{h}^\ell), \end{aligned} \quad (25.17)$$

where $\mathbf{h}^{L+1} \triangleq \mathbf{y}$ and $\mathcal{H} \cdot$ denotes the entropy. We note that within the ELBO derived using $\mathcal{Q}' = \prod_\ell p(\mathbf{f}^\ell | \mathbf{u}^\ell, \mathbf{h}^{\ell-1}) q(\mathbf{u}^\ell) q(\mathbf{h}^\ell)$, the per-layer term $\mathcal{L}^\ell$ arises as an exact computation — the $p(\mathbf{f}^\ell | \mathbf{u}^\ell, \mathbf{h}^{\ell-1})$ terms cancel between model and variational distribution — and the looseness relative to $\log p(\mathbf{h}^\ell | \mathbf{u}^\ell, \mathbf{h}^{\ell-1})$ is a consequence of using the prior conditional $p(\mathbf{f}^\ell | \mathbf{u}^\ell, \mathbf{h}^{\ell-1})$ rather than the posterior $p(\mathbf{f}^\ell | \mathbf{u}^\ell, \mathbf{h}^{\ell-1}, \mathbf{h}^\ell)$, a choice made for tractability.

Hence, all terms are tractable: the KL and entropy terms are standard Gaussian quantities, while the expectations involving covariance matrices require that the chosen kernel can be feasibly convolved with Gaussian densities, restricting the choice to kernels such as the exponentiated quadratic and the linear kernel. The bound factorises across data points, and is maximised with respect to model parameters $\{\beta^\ell, \boldsymbol{\theta}_f^\ell\}$ and variational parameters $\{\mathbf{x}_u^\ell, \mathbf{m}^\ell, \text{diag}(\mathbf{S}_i^\ell), \boldsymbol{\mu}^{\ell+1}, \boldsymbol{\Sigma}^{\ell+1}\}$. Notice that the variational parameters for $q(\mathbf{h}^\ell)$ scale as $\mathcal{O}(n)$ per layer.

Collapsed bound. Alternatively, one can determine the optimal $q(\mathbf{u}^\ell)$ analytically and substitute it back, yielding a *collapsed* bound that depends only on $q(\{\mathbf{H}^\ell\})$ [360]. This gives a tighter approximation, but the marginalisation of $q(\mathbf{u}^\ell)$ re-introduces coupling across data points: the collapsed objective no longer factorises in n , preventing minibatch training. For smaller datasets (where processing all data points jointly is feasible)

the collapsed bound may be preferred for its tightness; for larger datasets, the uncollapsed, factorised bound is necessary, and stochastic variational inference methods become essential (Section 25.3).

25.2.3 Supervised learning with noisy bottleneck layers

So far, we demonstrated the variational framework for the general scenario where all inputs are associated with a density, such as in an unsupervised learning scenario. However, in typical supervised learning applications, the top layer’s inputs are observed and deterministic. Therefore, inference for that layer is simpler, as there is no intractable density propagation issue; we explicitly model correlations in the input space through the approximate posterior $q(\mathbf{h}^1)$ which, in this case, constitutes a variational distribution q which is a Gaussian with *full* covariance matrix (rather than a diagonal one). Writing the general multi-dimensional form, we let $\mathbf{h}_j^\ell$ denote the j -th output dimension across all n datapoints (a column of $\mathbf{H}^\ell$) and $\mathbf{h}_{i,:}^\ell$ the i -th datapoint across all dimensions (a row of $\mathbf{H}^\ell$). The latent space distribution would then factorise as:

$$q(\{\mathbf{H}^\ell\}) = \prod_{j=1}^{d^1} \mathcal{N}(\mathbf{h}_j^1 | \mathbf{m}_j^1, \mathbf{S}_j^1) \prod_{\ell=2}^L \prod_{i=1}^n \mathcal{N}(\mathbf{h}_{i,:}^\ell | \mathbf{m}_{i,:}^\ell, \mathbf{S}_i^\ell), \quad (25.18)$$

where $\mathbf{S}_j^1$ is a *full* $n \times n$ matrix, as in Damianou et al. [364].

25.3 Scalable deep GPs

In section 25.2.2 we discussed two versions of the variational lower bound: collapsed and uncollapsed. For smaller training sets the collapsing approach might be preferred, because of the tighter bound and the reduced size of the parameters. For larger training sets, however, this approach becomes infeasible since all datapoints need to be considered together during the optimisation of the parameters. To tackle this issue, Dai et al. [357], Gal et al. [567] have devised parallel training regimes, while Dai et al. [358] takes this one step further by also amortising the variational distributions $q(\mathbf{h})$ such that their variational parameters do not scale per-datapoint but, instead, they are determined by a fixed, user-selected set of neural network variables; essentially, a multilayer perceptron (MLP) is trained to output these variables (see also Meng and Zhang [1165] for a recent treatment).

To scale to real-world big datasets, the researchers have further developed methodologies for stochastic variational inference (SVI) which allow the variational lower bound to handle training data as batches. These methodologies adopt the factorised, non-collapsed version of the variational bound, following the paradigm developed by Hensman et al. [737] for single-layer GPs. Indeed, since the likelihood factorises across data points, the bound involves the sparse GP conditional only through its per-datapoint marginals $p(f_n | \mathbf{u})$, with $f_n = f(x_n)$; the bound therefore decomposes into a sum over data points in which all dependence between data points is mediated by the global variables $\mathbf{u}$. Note that it is the structure of the bound, not of the model, that decouples the data points: the exact conditional $p(\mathbf{f} | \mathbf{u})$ itself remains correlated across data points through the residual covariance $\mathbf{K}$ of equation (25.14). Taking this one step further, Hensman and Lawrence [735] eliminate the variational distribution for the bottleneck layers entirely by marginalising via layer-wise compression. This allows for mini-batch optimisation, reduced variational parameter space as well as the incorporation of “messages” (in the form of Gaussian moments resulting from expectations of the marginalised $q(\mathbf{h})$) passed across layers.

Salimbeni and Deisenroth [1453] proposed *doubly stochastic variational inference* (DSVI), an approach in which no explicit $q(\mathbf{h}^\ell)$ distributions appear. Rather than propagating Gaussian messages across layers, DSVI retains the exact model conditional inside the approximate posterior and handles the resulting intractabilities through Monte Carlo sampling. A key departure from the framework of Sections 25.2.2–25.2.2 is in how layers are connected. Previously, each layer introduced a noise-corrupted bottleneck $\mathbf{h}^\ell = \mathbf{f}^\ell(\mathbf{h}^{\ell-1}) + \boldsymbol{\epsilon}^\ell$, and inference required estimating distributions over these noisy representations. In DSVI, the inter-layer noise is absorbed into the covariance function, so the GP function values $\mathbf{f}^\ell$ serve directly as the output of layer ℓ and input to layer $\ell + 1$. Concretely, this means the GP kernel at each layer includes a noise component $\sigma_\ell^2 \delta_{ij}$, but no separate latent variable $\mathbf{h}^\ell$ is introduced. Therefore, the distinction between function values $\mathbf{f}^\ell$ and noisy latent function instantiations $\mathbf{h}^\ell$ dissolves, and there is no separate $q(\mathbf{h}^\ell)$ to estimate.

Consider a deep GP with L layers, inputs $\mathbf{f}^0 = \mathbf{x}$, and each layer augmented with inducing variables $(\mathbf{u}^\ell, (\mathbf{x}_u)^\ell)$. The approximate posterior is:

$$\mathcal{Q}_{\text{DSVI}} = \prod_{\ell=1}^L p(\mathbf{f}^\ell | \mathbf{u}^\ell; \mathbf{f}^{\ell-1}) q(\mathbf{u}^\ell), \quad (25.19)$$

where $q(\mathbf{u}^\ell) = \mathcal{N}(\mathbf{u}^\ell | \boldsymbol{\mu}^\ell, \boldsymbol{\Sigma}^\ell)$ and $p(\mathbf{f}^\ell | \mathbf{u}^\ell; \mathbf{f}^{\ell-1})$ is the exact sparse GP conditional from the model. Since this conditional depends on $\mathbf{f}^{\ell-1}$ (the actual output of the previous layer, not a variational approximation to it), inter-layer dependencies are preserved. Since both factors are Gaussian, $\mathbf{u}^\ell$ can be marginalised analytically to yield:

$$q(\mathbf{f}^\ell | \mathbf{f}^{\ell-1}) = \int p(\mathbf{f}^\ell | \mathbf{u}^\ell; \mathbf{f}^{\ell-1}) q(\mathbf{u}^\ell) d\mathbf{u}^\ell = \mathcal{N}(\mathbf{f}^\ell | \tilde{\boldsymbol{\mu}}^\ell, \tilde{\mathbf{K}}_q^\ell), \quad (25.20)$$

where $\tilde{\boldsymbol{\mu}}^\ell$ and $\tilde{\mathbf{K}}_q^\ell$ take the same structural form as $\mathbf{a}^\ell$ and $\tilde{\mathbf{K}}^\ell$ in equation (25.14) but with $\mathbf{u}^\ell$ replaced by the variational mean $\boldsymbol{\mu}^\ell$ and the covariance adjusted to incorporate $\boldsymbol{\Sigma}^\ell$. The i -th marginal of this Gaussian depends only on $f_{i,:}^{\ell-1}$, since both its mean and its variance involve kernel evaluations of the layer inputs only at that point (together with the inducing inputs); as the bound below requires only these marginals, each data point can be propagated through the hierarchy independently.

In the ELBO, the model conditionals cancel between numerator and denominator, giving:

$$\mathcal{F}_{\text{DSVI}} = \sum_{i=1}^n \mathbb{E}_{q(f_{i,:}^L)} [\log p(y_{i,:} | f_{i,:}^L)] - \sum_{\ell=1}^L \text{D}_{\text{KL}}(q(\mathbf{u}^\ell) \| p(\mathbf{u}^\ell)). \quad (25.21)$$

The KL terms are analytic. The expected log-likelihood involves propagating each input $x_{i,:}$ through L layers of the stochastic conditionals (25.20), a composition of nonlinear transformations that is intractable in closed form. This is handled through two sources of stochasticity (hence the name): first, Monte Carlo sampling through the layers using the reparameterisation trick; second, minibatch subsampling over data points. Together, these yield unbiased gradient estimates of $\mathcal{F}_{\text{DSVI}}$.

Comparing with the bound of equation (25.17), the DSVI formulation differs in three main respects. First, the only variational parameters are the inducing distributions $q(\mathbf{u}^\ell)$, scaling as $\mathcal{O}(m^2)$ per layer regardless of n ; the $\mathcal{O}(n)$ parameters of $q(\mathbf{h}^\ell)$ are absent. The reduction in parameter count comes at the cost of not maintaining an explicit latent variable posterior over the intermediate representations, which the bound of equation (25.17) does provide. Second, no trace penalty $\frac{\beta^\ell}{2} \langle \text{tr}(\tilde{\mathbf{K}}^\ell) \rangle$ appears. In contrast to earlier approaches that rely on analytic expectations of kernels under Gaussian inputs, this formulation avoids such expectations and therefore does not impose corresponding restrictions on the choice of covariance function. Third, inter-layer correlations are retained: each layer's function values are conditioned on the sampled outputs of the previous layer rather than decoupled by a mean-field factorisation across layers. This reduces the variance underestimation typical of mean-field approximations, at the cost of the analytic tractability and closed-form per-layer bound that the factorisation affords. The first two differences reflect both the model structure (no separate noise variables) and the inference method (Monte Carlo rather than analytic expectations); the third reflects the inference method alone.

However, a mean-field assumption remains at the level of the inducing distributions $q(\mathbf{u}^\ell)$, which are treated as independent across layers; this factorisation can force intermediate layers to collapse to near-deterministic transformations, losing compositional uncertainty [1251, 1649]. In supervised regression and classification benchmarks, Salimbeni and Deisenroth [1453] demonstrated that deep GPs trained with DSVI consistently match or exceed single-layer GP performance, with the largest gains on large datasets — scaled up to a demonstration on over a billion points.

25.4 Further inference approaches

Sections 25.2.2 and 25.3 developed two variational inference frameworks in detail: the mean-field approach of Damianou and Lawrence [361] and the doubly stochastic formulation of Salimbeni and Deisenroth [1453]. These are not the only routes to approximate inference in deep GPs. In this section we survey three complementary approaches, each addressing different limitations of the variational schemes above. Together, they illustrate the trade-offs between approximation quality, computational cost, and modelling assumptions that shape practical choices.

25.4.1 Expectation propagation

Bui et al. [217] proposed an alternative to variational inference for deep GPs based on approximate expectation propagation (EP). Where variational methods optimise a lower bound on the marginal likelihood, EP approximates the posterior by iteratively matching the moments of local factors. Combined with inducing point representations and *stochastic* EP (which ties the local factors across data points for scalability), it propagates Gaussian approximations through successive layers, rather than maintaining an explicit variational distribution

over the hidden variables. On eleven regression benchmarks, Bui et al. [217] report that their method consistently outperforms single-layer GP regression and is competitive with Bayesian neural network baselines, while providing well-calibrated uncertainty estimates. A limitation is that EP requires moment computations under the approximate posterior that restrict the choice of kernel, similarly to the psi-statistic requirements of the [360, 1616] framework. See also Hernández-Muñoz et al. [744] for a more recent extension combining EP with Monte Carlo sampling.

25.4.2 Random feature approximations

A different strategy is to approximate each GP layer with a finite-dimensional feature expansion, turning the deep GP into a form of Bayesian neural network. Cutajar et al. [350] realise this at each layer using random feature expansions [1363], so that the deep GP composition takes the layer-wise form:

$$\mathbf{F}^{\ell+1} = \Phi^\ell \mathbf{W}^\ell, \quad \Phi^\ell = \gamma^\ell(\mathbf{F}^\ell \Omega^\ell), \quad (25.22)$$

up to the standard random-feature scaling, which we omit for clarity; in the above, $\mathbf{F}^\ell$ is the matrix of layer- ℓ outputs, $\mathbf{W}^\ell$ are Bayesian linear weights, the columns of Ω^ℓ are random frequencies sampled from a kernel-specific measure, and γ^ℓ is a kernel-specific nonlinearity.

Replacing each GP layer with this approximation transforms the deep GP into a deep network with random fixed features and Bayesian linear weights, on which stochastic variational inference can be performed in the standard weight-space manner. This approach is presented as substantially more scalable than inducing-point-based approaches and is demonstrated on datasets with over 8 million observations [350]. The trade-off is that the GP posterior at each layer is no longer represented explicitly in function space but rather through the weight distribution, and the feature expansion is restricted to kernels that admit a tractable random feature representation; Cutajar et al. [350] derive expansions for the squared exponential (via Bochner’s theorem, yielding trigonometric features) and for the arc-cosine kernel (via the integral representation of Cho and Saul [294], yielding ReLU features).

There is an appealing conceptual symmetry here with the constructions of Section 25.1: the chapter began by deriving deep GPs from deep neural networks via the infinite-width limit; random feature methods reverse this by approximating each GP layer with a finite-width neural network layer, returning to weight space for tractability.

25.4.3 Sampling-based and structured variational inference

The variational methods discussed so far, whether based on inducing points or random features, employ Gaussian approximate posteriors. Havasi et al. [714] argue that this can be a “potentially poor unimodal approximation of the generally multimodal posterior” and propose to bypass the Gaussian assumption entirely by applying stochastic gradient Hamiltonian Monte Carlo (SGHMC) (see Chapter 2) to sample from the posterior over the inducing outputs. Working within the inducing-point framework of Section 25.3, they interleave these SGHMC draws with hyperparameter optimisation via a novel moving window Monte Carlo EM algorithm. The resulting method achieves significantly better predictive performance at lower computational cost than variational inference on several regression benchmarks, and importantly, provides empirical evidence for the non-Gaussian nature of deep GP posteriors (see also Rossi et al. [1425], who extend this approach to jointly sample inducing inputs and hyperparameters). From a different angle, Dunlop et al. [463] propose a Metropolis-within-Gibbs sampler across the levels of the hierarchy. The sampler is demonstrated on function-space Bayesian inverse problems using a covariance-operator deep GP construction (distinct from the composition deep GPs that are the focus of the chapter). While neither sampler has been demonstrated at the very largest scales reached by the variational methods above, they establish quality benchmarks that expose fundamental limitations of Gaussian variational approximations.

Beyond switching inference paradigm, several works have sought to improve the fidelity of the variational family itself. Ustyuzhaninov et al. [1649] show that the standard mean-field approximation causes deep GP layers to collapse toward near-deterministic transformations, severely limiting uncertainty quantification; they propose a structured variational distribution that preserves inter-layer correlations. More recently, Xu et al. [1787] replace the Gaussian variational family with diffusion-based posteriors learned through a denoising process (cf. Chapter 50), shifting the paradigm from choosing a fixed-form parametric family to learning a flexible one.

In practice, DSVI [1453] remains a widely adopted inference method, with implementations available in frameworks such as GPflux [478] and GPyTorch [580]. The alternatives surveyed here offer complementary strengths: EP swaps variational ELBO optimisation for an EP energy function optimisation via moment matching, random features enable weight-space inference without matrix decompositions, and sampling methods and richer variational families relax the Gaussian posterior assumption itself.

25.5 Illustration

To further illustrate the properties and behaviour of deep GPs we consider a simple example, that is, attempting to fit the simulated step function; this follows the experimental setting of Calandra et al. [226], Hensman and Lawrence [735]. $n = 40$ equidistant training inputs $\{x_{i,:}\}_{i=1}^n$ were generated in the interval $[0, 1]$. As for the outputs, we set $y_{i,:} = 1$ if $x_{i,:} < 0.5$ and 0 otherwise, plus the addition of Gaussian noise with standard deviation 0.02. Therefore, the output space is very “flat” but broken by a big discontinuity (step), thus making the application of a standard GP very challenging. In contrast, a deep GP as a generative model is able to capture this extreme non-stationarity and bimodality by making use of the available intermediate layers that gradually “push” the points in the two extrema. The results presented below confirm this, and they are very similar to those of Hensman and Lawrence [735] who used a deep GP variant with nested variational compression.

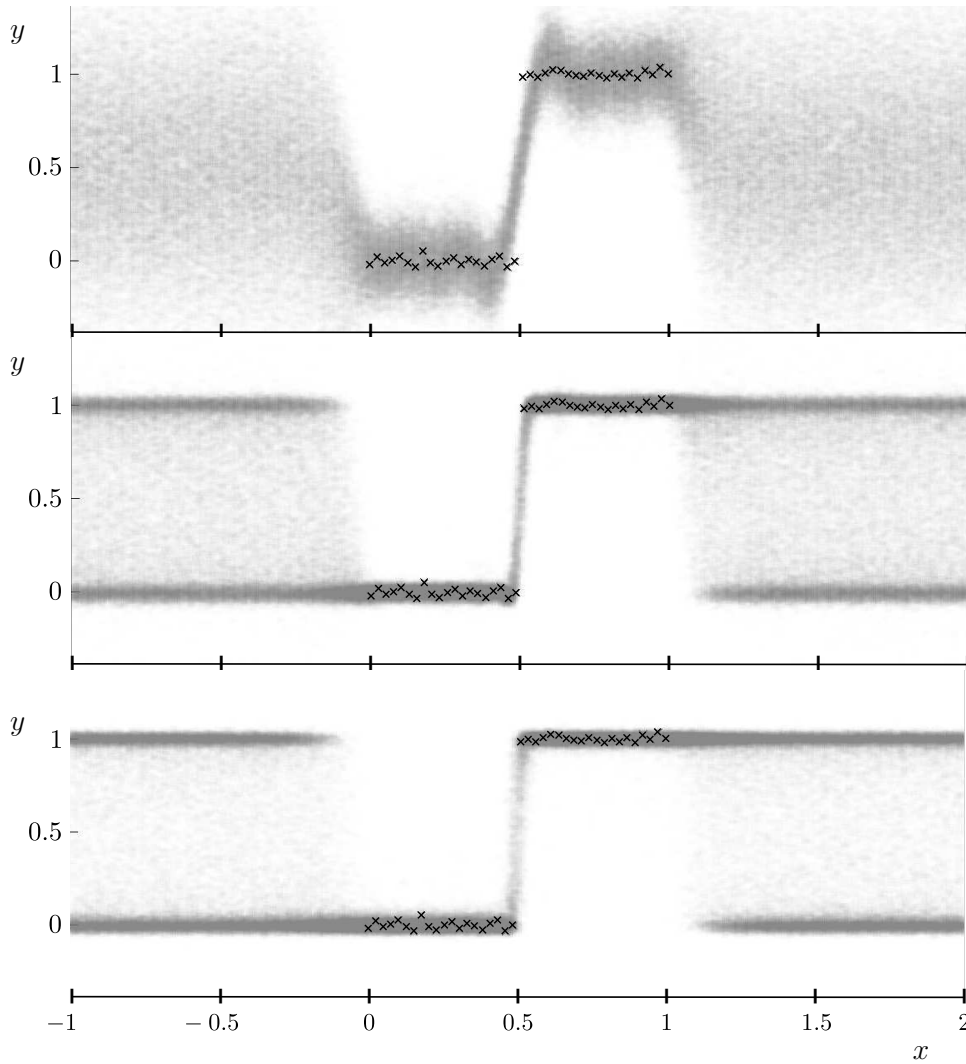
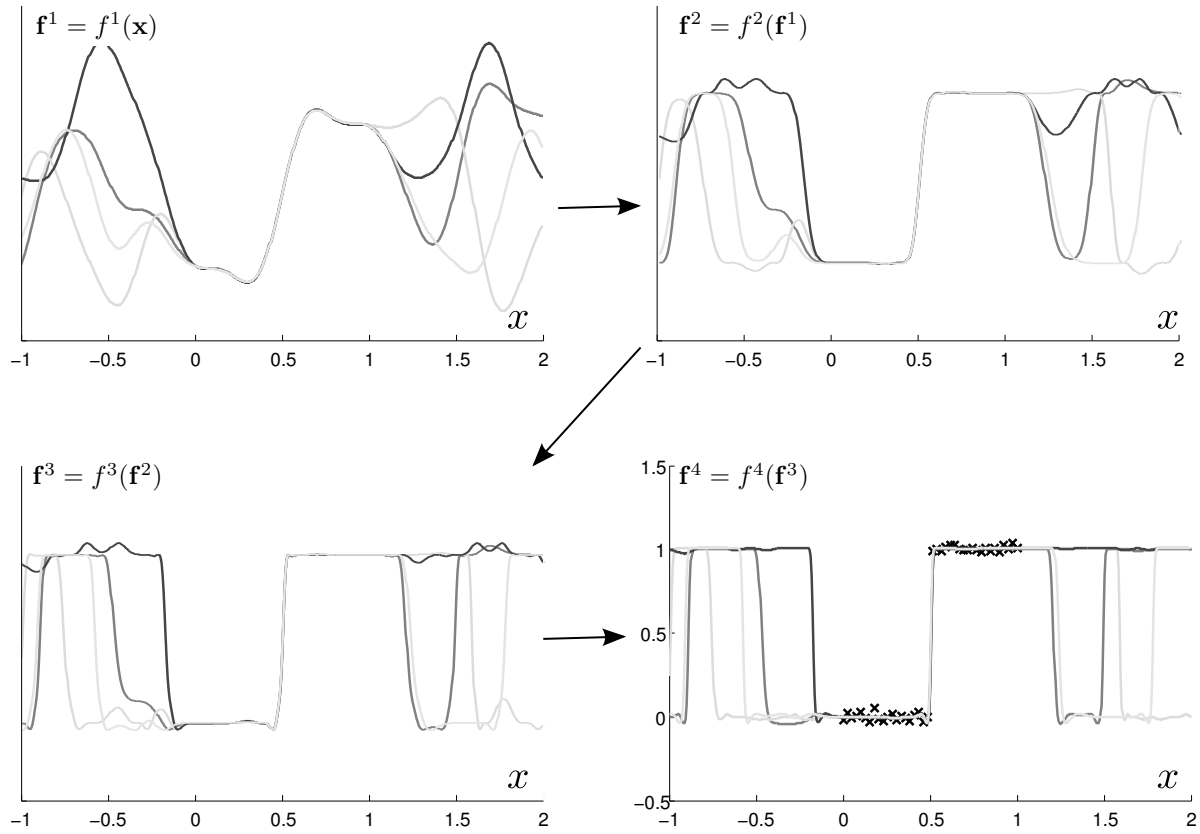
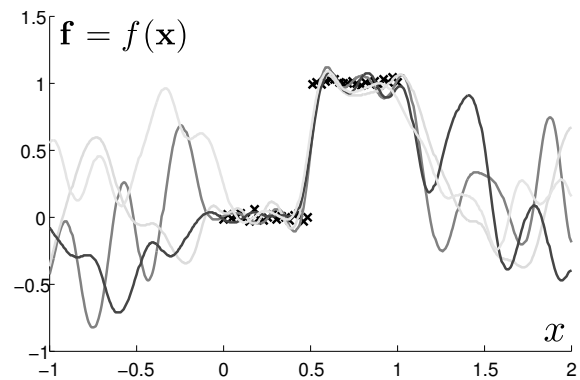


Figure 25.6: Samples drawn from the posterior of a standard GP (top), a deep GP with 1 hidden layer (middle) and a deep GP with three hidden layers (bottom). As in Hensman and Lawrence [735], for better visualisation the sampled points are drawn as small, faint semi-transparent squares, whereas the training points are plotted as dark opaque x’s.

Indeed, Figure 25.6 shows 300 sample paths from the posterior obtained by a standard GP with exponentiated quadratic covariance function (top panel) and a deep GP with one (middle panel) and three (bottom panel) hidden layers. Each of the paths corresponds to the discretisation of the interval $[-1, 2]$ in 500 points. The figure demonstrates how even a 1 hidden layer deep GP is able to easily operate in a regime which seems bimodal.



(a) Deep GP with three hidden plus one input-transformation layer



(b) Standard GP

Figure 25.7: (a) Successive warping of sample paths drawn from a deep GP, gradually “pushing” them towards a bimodal regime. (b) Sample paths drawn from a standard GP. In this figure, each sample is drawn versus the initial input $\mathbf{x}$. Instead, figure 25.8 shows a plot of each layer’s output signal versus its input signal.

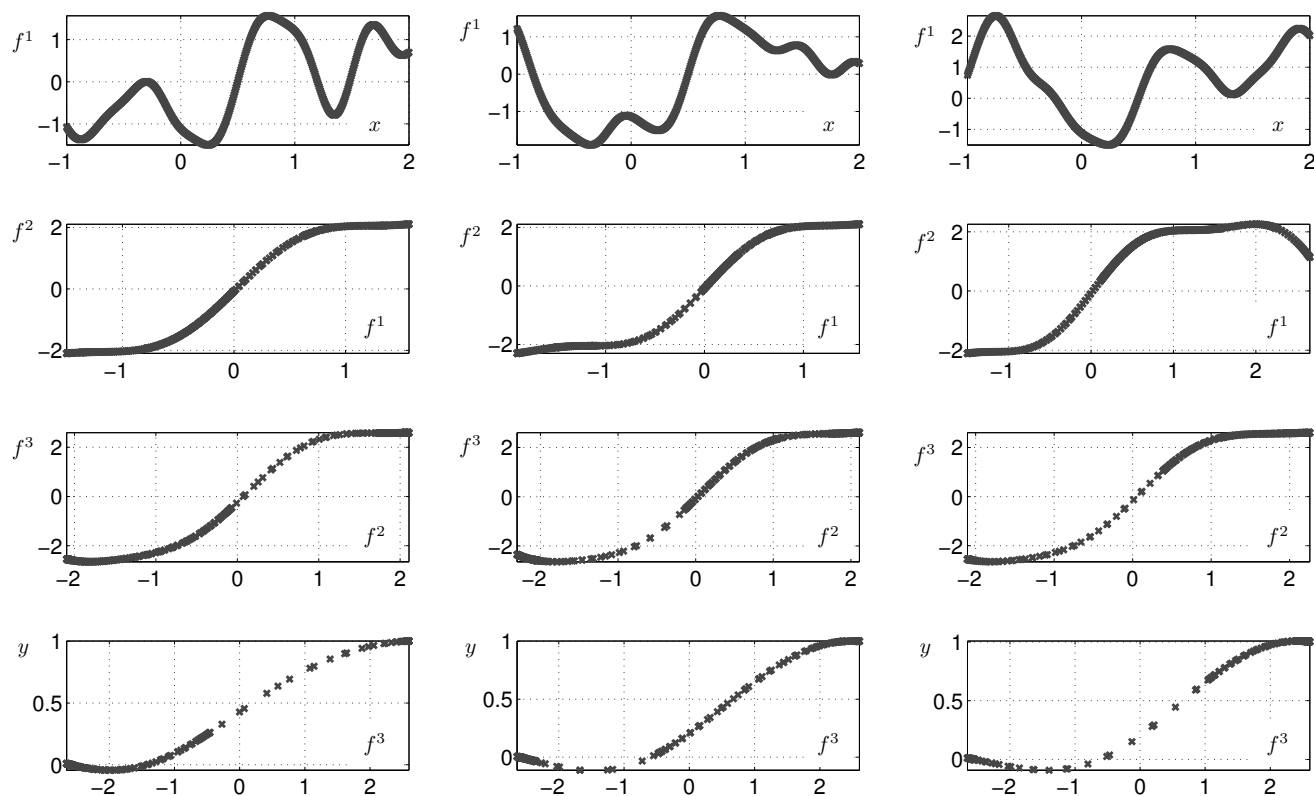


Figure 25.8: Plotting the input versus the output sample signals of every layer in the deep GP. This reveals sigmoidal warping functions which successively “push” the top layer’s samples into the bi-modal regime. In the axis labels, the superscript denotes the layer number; each column corresponds to a single sample propagated through the hierarchy.

A deeper architecture (bottom panel) fits in this regime even better, with fewer samples falling between the two extrema. It is interesting to examine the locations of these samples: they seem to cover uniformly and very sparsely the area between the “modes”. Further, such samples are only obtained away from the interval $[0, 1]$, signifying that a “jump” is expected to happen (although with extremely low probability) in regions away from the data. On the other hand, away from the data the standard GP produces many samples corresponding to y values that are too far from the training points. Furthermore, the GP is even forced to underfit the data; there are no samples falling in the region close to the “jump” even if two training points (one up and one down) exist there while, at the same time, the variance around the training data is too large.

To confirm the above intuitions, as well as to draw further conclusions, in Figure 25.7 we plot whole sample paths from the deep and the standard GP. For the deep GP, Figure 25.7(a) shows the samples’ successive warping through the deep hierarchy, which gradually “pushes” them towards the bimodal regime. As can be seen, the samples (especially in the middle layers) are smooth, and are probabilistically assigned to one of the two modes. Jumps can occur but are rare, perhaps because only one jump exists in the training data. The interpolation suggests that the deep GP interprets the differences of the points *within* each cluster mostly as noise, which matches the actual toy data generation. Finally, Figure 25.8 also depicts sample paths, but there each plot shows the sample output signal in every layer versus its corresponding *input signal* – in contrast to Figure 25.7 where the x -axis is the input $\mathbf{x}$. As can be seen in Figure 25.8, each layer’s warping function is learned to have a sigmoidal shape, so as to successively “push” the top layer’s samples into the bi-modal regime.

Chapter 26

Deep kernel processes and deep kernel machines

26.1 Introduction

Neural networks have become the default choice for complex tasks such as computer vision and natural language processing due to their huge performance advantage over other methods [74, 213, 445, 721, 748, 935, 1361, 1782]. Due to their black-box nature, neural network design is almost entirely based on trial-and-error, and once deployed, it is very difficult to determine what factors contribute to a neural network’s decisions; it is often found that neural networks are using unexpected and extraneous features for their predictions [444, 1395]. To exacerbate the issue, our understanding of deep learning theory currently lags far behind practice [e.g. 110, 1407, 1795, 1842]. This leads to models that, although powerful, are time-consuming and finicky to design (e.g. compare He et al. [721] and He et al. [722], or consider the fact that neural architecture search is its own extensive research area [491]), often more computationally expensive than necessary [411, 749, 1588], and potentially unsuitable for use in safety-critical applications like medicine and autonomous systems [30, 499, 1758].

Kernel methods are a class of models that offer strong interpretability [492, 798, 1341, 1863], since their operation explicitly computes similarities between datapoints. Furthermore, many theoretical models of neural networks reduce to kernel methods, allowing a cleaner analysis [386, 829, 1000, 1792]. Unfortunately, kernel methods largely fall short of neural networks in predictive performance on many important machine learning tasks.

Recent developments in kernel method are capable of fixing this shortfall [16, 1792]. These are Deep Kernel Processes (DKPs), and Deep Kernel Machines (DKMs).

26.2 Background

In this chapter we provide the necessary background to understand DKPs and DKMs. We first recap Gaussian processes, before introducing the approximate sparse (inducing point) Gaussian processes that underlie DKPs/DKMs, as well as Deep Gaussian Processes (DGPs). We then introduce deep kernel processes, which are a “no-feature” view DGPs, and deep kernel machines, which are the infinite-width version of a DKPs. Note that the notation of this chapter differs from that of other chapters.

26.2.1 Gaussian Processes

Whilst neural networks have shown impressive abilities to learn features and generalise on complex tasks [935], the precise reasons for why they work so well in these settings is poorly understood, and in many scenarios, such as those with small amounts of data, neural networks can struggle compared to more traditional methods. Hence in this section, we recap Gaussian processes (or GPs) from Bayesian statistics, which are a type of Bayesian kernel method, and in later sections will explore the relationship between neural networks and Gaussian processes. Gaussian processes were previously introduced in Chapter 23.

A Gaussian process [1382] is a stochastic process (a collection of random variables) where any finite subset is multivariate Gaussian distributed. A Gaussian process is completely specified by its mean function $m(\cdot) \in \mathbb{R}^d \rightarrow \mathbb{R}$

and positive-definite covariance / kernel function $k(\cdot, \cdot) \in \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, for which we write,

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)). \quad (26.1)$$

Gaussian processes are used to represent prior distributions over functions. Concretely, if $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_0}$ are two datapoints of dimensionality N_0 , then for a function $f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$, we have,

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \quad (26.2)$$

$$\mathbb{E}[f(\mathbf{x}')] = m(\mathbf{x}') \quad (26.3)$$

$$\text{cov}(f(\mathbf{x}), f(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}'). \quad (26.4)$$

In particular, if we have P datapoints $\{\mathbf{x}_1, \dots, \mathbf{x}_P\}$, we have that the joint distribution of the function values evaluated at those points is multivariate Gaussian:

$$\begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_P) \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_P) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_P) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_P, \mathbf{x}_1) & \cdots & k(\mathbf{x}_P, \mathbf{x}_P) \end{pmatrix} \right). \quad (26.5)$$

Hence we can see that the mean function $m(\cdot)$ tells us the average / expected value of the function at a particular input point, and the kernel function $k(\cdot, \cdot)$ tells us how the function value at a particular point is influenced by the function value at nearby points. Probably the most intuitive class of kernels are the isotropic kernels, which can be computed as a function of the distance between two points, i.e. $k(\|\mathbf{x} - \mathbf{x}'\|)$. A very common example is the squared exponential kernel,

$$k(\mathbf{x}, \mathbf{x}') = \exp \left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2} \right), \quad (26.6)$$

where l is a hyperparameter known as the “lengthscale”, which controls how quickly covariances decay with distance (a bigger lengthscale represents smoother functions that capture broad trends, whilst a smaller lengthscale represents rapidly varying functions that capture local information). There are many other types of kernel function, such as periodic kernels whose covariance oscillates with distance, giving periodic functions, which we will not discuss here. In many cases, we don’t have any prior information about the mean of the function, so the mean function $\mathbf{m}(\cdot)$ will often be set to the zero function $\mathbf{m}(\mathbf{x}) = \mathbf{0}$ for all $\mathbf{x} \in \mathbb{R}^{N_0}$.

Assuming we have targets $\{y_1, \dots, y_P\}$ and a likelihood function $p(y_i | f(x_i))$, with a Gaussian process prior, we can perform Bayesian inference for **non-parametric** (or “function-space”) function approximation. This is in contrast to **parametric** / weight-space methods, which assume a parametric form for a function like $f(x_i) = w_1 x_i + w_2$ in Bayesian linear regression, and compute a posterior over the weights $p(w_1, w_2 | \{x_1, \dots, x_P\}, \{y_1, \dots, y_P\})$ (which itself could then be used to compute a predictive distribution).

Specifically, given a new set of test points $\{\mathbf{x}_1^*, \dots, \mathbf{x}_{P_*}^*\}$, we wish to compute the posterior $p(f(\mathbf{x}_1^*), \dots, f(\mathbf{x}_{P_*}^*) | \{x_1, \dots, x_P\})$ over the value of the function at the new test point. This is made possible in certain cases due to the closed-form “partitioned Gaussian” formulae. Suppose I have a multivariate Gaussian random vector which is partitioned into two shorter vectors $\mathbf{a}$ and $\mathbf{b}$ in the following way:

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix} \right). \quad (26.7)$$

Then the conditional distribution $p(\mathbf{b} | \mathbf{a})$ is also multivariate Gaussian, and has the form,

$$\mathbf{b} | \mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_b + \boldsymbol{\Sigma}_{ba} \boldsymbol{\Sigma}_{aa}^{-1}(\mathbf{a} - \boldsymbol{\mu}_a), \boldsymbol{\Sigma}_{bb} - \boldsymbol{\Sigma}_{ba} \boldsymbol{\Sigma}_{aa}^{-1} \boldsymbol{\Sigma}_{ab}). \quad (26.8)$$

Assuming noiseless observations $y_i = f(x_i)$ (i.e. we directly observe the latents $\mathbf{f}$), we can form a partitioned multivariate Gaussian prior using our train points $\{f(\mathbf{x}_1), \dots, f(\mathbf{x}_P)\}$ and our test points $\{f(\mathbf{x}_1^*), \dots, f(\mathbf{x}_{P_*}^*)\}$ as follows:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mathbf{m}(\mathbf{X}) \\ \mathbf{m}(\mathbf{X}_*) \end{pmatrix}, \begin{pmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{pmatrix} \right) \quad (26.9)$$

where $\mathbf{X} \in \mathbb{R}^{P \times N_0}$, $\mathbf{X}_* \in \mathbb{R}^{P_* \times N_0}$ are the vertically stacked inputs and $\mathbf{f} \in \mathbb{R}^P$, $\mathbf{f}_* \in \mathbb{R}^{P_*}$ are the vertically stacked function values evaluated at those inputs, $\mathbf{m}(\mathbf{X})$, $\mathbf{m}(\mathbf{X}_*)$ are the vertically stacked mean values for those inputs, and $\mathbf{K}(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{P \times P}$, $\mathbf{K}(\mathbf{X}, \mathbf{X}_*) \in \mathbb{R}^{P \times P_*}$, $\mathbf{K}(\mathbf{X}_*, \mathbf{X}) \in \mathbb{R}^{P_* \times P}$, $\mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \in \mathbb{R}^{P_* \times P_*}$ are the “kernel matrices”, formed by applying the kernel function $k(\cdot, \cdot)$ to all pairs of train vs. train, train vs. test, test vs. train, and

test vs. test points, respectively. Note that since $k(\cdot, \cdot)$ is a symmetric positive-definite function, $\mathbf{K}(\mathbf{X}, \mathbf{X})$ and $\mathbf{K}(\mathbf{X}_*, \mathbf{X}_*)$ are symmetric positive-definite matrices, and $K(\mathbf{X}, \mathbf{X}_*) = K(\mathbf{X}_*, \mathbf{X})^T$.

Applying the partitioned Gaussian formula allows us to compute our posterior predictive distribution as,

$$\mathbf{f}_* | \mathbf{f} \sim \mathcal{N}(\mathbf{m}_* + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}), \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*), \quad (26.10)$$

where we used the abbreviations $\mathbf{m} = \mathbf{m}(\mathbf{X})$, $\mathbf{m}_* = \mathbf{m}(\mathbf{X}_*)$, $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = \mathbf{K}(\mathbf{X}, \mathbf{X}_*)$, $\mathbf{K}_{**} = \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*)$. This is common in the Gaussian process literature. Hence we have a closed-form solution for the distribution of our test predictions $\mathbf{f}_*$. Note it is common to drop the conditioning on the inputs $\mathbf{X}$.

We also get a closed-form solution when we assume Gaussian noise on our data, i.e. $y_i = f(\mathbf{x}_i) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2 \in \mathbb{R}^+$. Since the sum of two Gaussians is itself Gaussian, the joint distribution over training labels $\mathbf{y} = \mathbf{f} + \epsilon$ and test labels $\mathbf{y}_* = \mathbf{f}_* + \epsilon$ (where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$) becomes

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{y}_* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mathbf{m}(\mathbf{X}) \\ \mathbf{m}(\mathbf{X}_*) \end{pmatrix}, \begin{pmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) + \sigma^2 \mathbf{I} \end{pmatrix} \right). \quad (26.11)$$

We can form the posterior predictive distribution

$$\mathbf{y}_* | \mathbf{y} \sim \mathcal{N}(\mathbf{m}_* + \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}), \mathbf{K}_{**} - \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_* + \sigma^2 \mathbf{I}). \quad (26.12)$$

In the simple case of Gaussian noise regression, it is not uncommon to see texts present “prediction” as predicting the underlying latent $\mathbf{f}_*$, so we actually form the joint distribution between $\mathbf{y}$ and $\mathbf{f}_*$:

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mathbf{m}(\mathbf{X}) \\ \mathbf{m}(\mathbf{X}_*) \end{pmatrix}, \begin{pmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{pmatrix} \right), \quad (26.13)$$

and hence we can compute the posterior over $\mathbf{f}_*$ as

$$\mathbf{f}_* | \mathbf{y} \sim \mathcal{N}(\mathbf{m}_* + \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}), \mathbf{K}_{**} - \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_*). \quad (26.14)$$

In summary, we assumed a multivariate Gaussian prior $p(\mathbf{f})$ over our latents, and a Gaussian likelihood $p(\mathbf{y} | \mathbf{f}) = \mathcal{N}(\mathbf{y}; \mathbf{f}, \sigma^2 \mathbf{I})$.

For completeness and to intuitively motivate GPs, we can think of a GP as an infinite-dimensional multivariate Gaussian, where the dimensions represent the function value at all possible points $\mathbf{x}$ in the input space. Specifically, it retains the property that all marginal and conditional distributions over subsets of variables are also Gaussian. Thus, the prior at a finite set of input locations $\mathbf{X}$ is a finite-dimensional multivariate Gaussian $p(\mathbf{f})$, and the “prior conditional” distribution at a set of finite set of input locations $\mathbf{X}_*$ given the known function values at a different set of inputs $\mathbf{X}$ is a multivariate Gaussian $p(\mathbf{f}_* | \mathbf{f})$, given by the conditional Gaussian formulae in Eq. 26.8. In the case of a Gaussian likelihood function, **the posterior process is also a Gaussian process**, i.e. the posterior distribution over the entire function $f(\cdot)$ has the form

$$f(\cdot) | X_{\text{data}}, y_{\text{data}} \sim \mathcal{GP}(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot)) \quad (26.15)$$

where

$$m_{\text{post}}(x) = m(x) + k(x, X_{\text{data}}) [K(X_{\text{data}}, X_{\text{data}}) + \sigma_n^2 I]^{-1} (y_{\text{data}} - m(X_{\text{data}})) \quad (26.16)$$

$$k_{\text{post}}(x, x') = k(x, x') - k(x, X_{\text{data}}) [K(X_{\text{data}}, X_{\text{data}}) + \sigma_n^2 I]^{-1} k(X_{\text{data}}, x'). \quad (26.17)$$

That is, $m(\cdot)$, $k(\cdot, \cdot)$ are our prior mean and kernel function, and $m_{\text{post}}(\cdot)$, $k_{\text{post}}(\cdot, \cdot)$ are our posterior mean and kernel function. This is what makes the Gaussian process ‘non-parametric’. The posterior is really over the infinite-dimensional Gaussian process, but in practice, we can only manipulate expressions involving finite sets of locations $\mathbf{X}$ and $\mathbf{X}_*$. As we will see in the next section, the posterior process is only a Gaussian process in the case of a Gaussian likelihood. When using a non-Gaussian likelihood (such as for classification), we must resort to approximations.

26.2.2 Variational Inference for Gaussian Process Classification

Variational inference is discussed in depth in Chapter 12, but we provide a self-contained recap here. Suppose we have data $\mathbf{x}_i \in \mathbb{R}^{N_0}$, with binary labels $y_i \in \{0, 1\}$. A Gaussian likelihood function is not appropriate here, as

our labels take discrete values, whereas a Gaussian distribution has continuous and unbounded support. We work around this by using a logistic / sigmoid function to form a Bernoulli likelihood by transforming the latents,

$$p(y_i = 1|f_i) = \sigma(f_i) = \frac{1}{1 + \exp(-f_i)}. \quad (26.18)$$

More generally, if $y_i \in \{0, \dots, K\}$ are labels in a K -class classification task, we can use a Categorical likelihood with softmax probabilities computed from a vector latent $\mathbf{f}(\cdot) \in \mathbb{R}^K$,

$$p(y_i = k|\mathbf{f}_i) = \text{softmax}(\mathbf{f}_i)_k = \frac{\exp(f_{ik})}{\sum_{k'} \exp(f_{ik'})}. \quad (26.19)$$

We immediately encounter a problem: given a multivariate Gaussian prior $p(\mathbf{f})$ over the latents, and a Bernoulli likelihood $p(\mathbf{y}|\mathbf{f})$, the posterior over the latents $p(\mathbf{f})$ is not closed-form. Even if it were closed-form, but not Gaussian, then the integral required to compute the predictive distribution $p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*)p(\mathbf{f}_*|\mathbf{y})d\mathbf{f}_*$ almost certainly does not have a closed form solution.

Hence approximation is necessary. There are many ways we could go about this, but here we consider using variational inference to obtain an approximate posterior $q(\mathbf{f}_*)$, and then use samples $\mathbf{f}_{*,(m)}$ from this approximate posterior to compute the Monte-Carlo average

$$p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*)p(\mathbf{f}_*|\mathbf{y})d\mathbf{f}_* \approx \frac{1}{M} \sum_m \sigma(\mathbf{f}_{*,(m)}) \quad \mathbf{f}_{*,(m)} \in \mathbb{R}^{P_*} \sim q(\mathbf{f}_*). \quad (26.20)$$

In variational inference [170, 851, 1691], we seek to approximate an intractable posterior $p(\mathbf{f}|\mathbf{y})$ with an approximate posterior $q(\mathbf{f})$ which is easier to manipulate and sample from. A very common choice, is a multivariate Gaussian approximate posterior $q(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ parametrised by learnable mean and covariance matrix $\boldsymbol{\mu}_q \in \mathbb{R}^P$, $\boldsymbol{\Sigma}_q \in \mathbb{R}^{P \times P}$. In VI, q is called a ‘variational approximate posterior’, and $\boldsymbol{\mu}_q$ and $\boldsymbol{\Sigma}_q$ are ‘variational parameters’. The main benefit of this approximation is that it allows closed-form expressions for the approximate predictive posterior $q(\mathbf{f}_*)$.

If $q(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ is our approximate posterior over the training latents, then a sensible choice of the approximate posterior over the test latents $\mathbf{f}_*$ is given by the integral,

$$q(\mathbf{f}_*) = \int p(\mathbf{f}_*|\mathbf{f})q(\mathbf{f})d\mathbf{f}. \quad (26.21)$$

This is sensible because it closely resembles the true posterior,

$$p(\mathbf{f}_*) = \int p(\mathbf{f}_*, \mathbf{f})d\mathbf{f} = \int p(\mathbf{f}_*|\mathbf{f})p(\mathbf{f})d\mathbf{f}. \quad (26.22)$$

Since $p(\mathbf{f}_*|\mathbf{f}) = \mathcal{N}(\mathbf{f}_*; \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*)$ and $q(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ are both Gaussian, it is a standard result that their integral will also be Gaussian, and we can compute the mean and variance using the laws of total expectation and total variance:

$$\mathbb{E}_{q(\mathbf{f})}[\mathbf{f}_*] = \mathbb{E}_{q(\mathbf{f})}[\mathbb{E}[\mathbf{f}_*|\mathbf{f}]] \quad (26.23)$$

$$= \mathbb{E}_{q(\mathbf{f})}[\mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f}] \quad (26.24)$$

$$= \mathbf{K}_*^T \mathbf{K}^{-1} \boldsymbol{\mu}_q \quad (26.25)$$

$$\text{Var}_{q(\mathbf{f})}[\mathbf{f}_*] = \mathbb{E}_{q(\mathbf{f})}[\text{Var}[\mathbf{f}_*|\mathbf{f}]] + \text{Var}_{q(\mathbf{f})}[\mathbb{E}[\mathbf{f}_*|\mathbf{f}]] \quad (26.26)$$

$$= \mathbb{E}_{q(\mathbf{f})}[\mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*] + \text{Var}_{q(\mathbf{f})}[\mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f}] \quad (26.27)$$

$$= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* + \mathbf{K}_*^T \mathbf{K}^{-1} \text{Var}_{q(\mathbf{f})}[\mathbf{f}] \mathbf{K}^{-1} \mathbf{K}_* \quad (26.28)$$

$$= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* + \mathbf{K}_*^T \mathbf{K}^{-1} \boldsymbol{\Sigma}_q \mathbf{K}^{-1} \mathbf{K}_*. \quad (26.29)$$

Note that for clarity we have used subscript $q(\mathbf{f})$ in the expectations and variances which integrate over the approximate posterior, and omitted the subscripts in expressions that only involve the prior $p(\mathbf{f}_* | \mathbf{f})$. Hence, assuming we have values for $\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q$, we can generate samples from the approximate predictive distribution

$$q(\mathbf{f}_*) = \mathcal{N}(\mathbf{f}_*; \mathbf{K}_*^T \mathbf{K}^{-1} \boldsymbol{\mu}_q, \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* + \mathbf{K}_*^T \mathbf{K}^{-1} \boldsymbol{\Sigma}_q \mathbf{K}^{-1} \mathbf{K}_*) \quad (26.30)$$

and then compute Monte-Carlo averaged predictions for $p(\mathbf{y}_*|\mathbf{y})$ as in Eq. (26.20).

All that remains is to determine the values for $\mu_q \in \mathbb{R}^P$, $\Sigma_q \in \mathbb{R}^{P \times P}$, that make our approximate posterior $q(\mathbf{f})$ as close to the true posterior $p(\mathbf{f}|\mathbf{y})$ as possible. In variational inference, the desired measure of “closeness” is the KL divergence:

$$\mu_q, \Sigma_q = \arg \min_{\mu, \Sigma} D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y})) = \arg \min_{\mu, \Sigma} \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f}|\mathbf{y})} d\mathbf{f}. \quad (26.31)$$

At first glance, this seems difficult because the true posterior $p(\mathbf{f}|\mathbf{y})$ is unknown. However, we can use a fairly common trick of rewriting the divergence in terms of the ELBO (Evidence Lower Bound Objective),

$$D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y})) \quad (26.32a)$$

$$= \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f}|\mathbf{y})} d\mathbf{f} \quad (26.32b)$$

$$= \int q(\mathbf{f}) \log q(\mathbf{f}) d\mathbf{f} - \int q(\mathbf{f}) \log p(\mathbf{f}|\mathbf{y}) d\mathbf{f} \quad (26.32c)$$

$$= \int q(\mathbf{f}) \log q(\mathbf{f}) d\mathbf{f} - \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y})} d\mathbf{f} \quad (26.32d)$$

$$= \int q(\mathbf{f}) [\log q(\mathbf{f}) - \log p(\mathbf{f})] d\mathbf{f} - \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \int q(\mathbf{f}) \log p(\mathbf{y}) d\mathbf{f} \quad (26.32e)$$

$$= \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} - \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log p(\mathbf{y}) \int q(\mathbf{f}) d\mathbf{f} \quad (26.32f)$$

$$= \underbrace{D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}))}_{\text{Negative ELBO}} - \underbrace{\mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}|\mathbf{f})]}_{\text{Evidence}} + \underbrace{\log p(\mathbf{y})}_{\text{Evidence}}. \quad (26.32g)$$

Rearranging, in terms of the evidence,

$$\log p(\mathbf{y}) = D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y})) + \underbrace{\mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}|\mathbf{f})] - D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}))}_{\text{ELBO}}. \quad (26.33)$$

The divergence between the approximate and true posteriors, $D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y}))$, is always positive, so the ELBO acts as a lower bound to the model evidence. When the data and model are fixed, the evidence is constant, and so by maximising the ELBO, we simultaneously minimise the KL divergence between the approximate posterior and the true posterior, which is our overall goal. To see whether we can optimise the ELBO,

$$\text{ELBO} = \underbrace{\mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}|\mathbf{f})]}_{\text{Expected Log Likelihood (under } q)} - \underbrace{D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f}))}_{\text{KL between approx. post. and prior}}. \quad (26.34)$$

Observe that the first term is the expected log-likelihood under the approximate posterior $q(\mathbf{f})$. As suggested earlier, we can estimate this term by taking samples from $q(\mathbf{f})$. Since we wish to differentiate the ELBO in order to maximise it with gradient-ascent, we will use the reparametrisation trick [892] to detach the random sampling process from the computation graph. That is, our Gaussian random sample $\mathbf{f}$ will be written as a transformation of a standard Gaussian sample, so that all learnable parameters are in the deterministic transformation, rather than random variables,

$$\mathbf{f} = \mu_q + \mathbf{L}\mathbf{z} \quad \text{where } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ and } \Sigma = \mathbf{L}\mathbf{L}^T. \quad (26.35)$$

The second term is the KL divergence between the approximate posterior and the prior over $\mathbf{f}$. Since these are both Gaussian ($p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$ and $q(\mathbf{f}) = \mathcal{N}(\mu_q, \Sigma_q)$), this can be computed in closed form:

$$D_{\text{KL}}(q(\mathbf{f})||p(\mathbf{f})) = \frac{1}{2} \left[\log \frac{|\mathbf{K}|}{|\Sigma_q|} - P + \mu_q^T \mathbf{K}^{-1} \mu_q + \text{tr}(\mathbf{K}^{-1} \Sigma_q) \right] \quad (26.36)$$

where $|\mathbf{A}|$ denotes the determinant of a matrix $\mathbf{A}$, and $\text{tr}(\mathbf{A})$ the trace, and as a reminder, P is the number of training datapoints, dictating the size of $\mathbf{K} \in \mathbb{R}^{P \times P}$.

As a final point, we note that since the ELBO is a lower bound to the model evidence, it is common to optimise model hyperparameters (i.e. hyperparameters of the prior $p(\mathbf{f})$) by maximising the ELBO as a proxy to the model evidence.

Now that we’ve seen how we can use variational inference to do GP classification, let’s discuss how practical (or, rather, impractical) it is. The key problem is the dependence on the dataset size P . The kernel matrix $\mathbf{K} \in \mathbb{R}^{P \times P}$ and the approximate posterior covariance $\Sigma_q \in \mathbb{R}^{P \times P}$ both have P^2 elements, where P is the number

of training datapoints. To store Σ_q in memory for moderately sized dataset with $P = 10^6$ datapoints using 32-bit floating point numbers would require $(10^6)^2 \times 4$ bytes, which equates to roughly 3,725GB of memory. Supposing that memory were not a concern, we must still compute expensive operations on the kernel matrix, such as its inverse $\mathbf{K}^{-1}$ and its determinant $|\mathbf{K}|$. Both of these operations are very expensive and are impractical for large P .

In the next section, we introduce “sparse” variational Gaussian processes, which replace the P training points with P_i “inducing points” which seek to summarise the datapoint, but with $P_i \ll P$ so that we get practical $\mathcal{O}(P_i^3)$ operations.

26.2.3 Sparse Variational Gaussian Processes

Sparse Gaussian processes [737, 738, 1354, 1528, 1615] seek to alleviate the $\mathcal{O}(P^2)$ space complexity and $\mathcal{O}(P^3)$ time complexity of Gaussian processes (where P is the number of training points) by replacing the training points with a small set of $P_i \ll P$ “inducing points” that somehow summarise the dataset. These inducing points may simply be a subset of the original training data, or they may be learnable parameters, optimised in some way. Here, we follow the variational inducing point approach as introduced by Titsias [1615] and refined by Hensman et al. [737], where inducing points are inserted via the approximate posterior.

Concretely, let $\mathbf{X}_i \in \mathbb{R}^{P_i \times N_0}$ be a set of “inducing inputs”, that is, a set of P_i input datapoints, and let $\mathbf{f}_i \in \mathbb{R}^{P_i}$ be the “inducing values”, i.e. function values evaluated at those inducing inputs. Furthermore, for absolute clarity, we introduce new notation $\mathbf{f}_t \in \mathbb{R}^{P_t}$ for our set of P_t training points.

As always, the joint distribution between function evaluations is multivariate Gaussian due to our Gaussian process prior. This means the joint prior over our inducing points $\mathbf{f}_i$ and our train points $\mathbf{f}_t$ is multivariate Gaussian:

$$\mathbf{f} = \begin{pmatrix} \mathbf{f}_i \\ \mathbf{f}_t \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{ii} & \mathbf{K}_{it} \\ \mathbf{K}_{ti} & \mathbf{K}_{tt} \end{pmatrix} \right). \quad (26.37)$$

Here, $\mathbf{f} \in \mathbb{R}^{P_i+P_t}$ denotes the concatenation of our inducing and train points, $\mathbf{K}_{ii} = \mathbf{K}(\mathbf{X}_i, \mathbf{X}_i) \in \mathbb{R}^{P_i \times P_i}$ is the kernel matrix formed by applying the kernel to all pairs of inducing points, $\mathbf{K}_{ti} = \mathbf{K}_{it}^T = \mathbf{K}(\mathbf{X}_t, \mathbf{X}_i) \in \mathbb{R}^{P_t \times P_i}$ is the kernel matrix formed by applying the kernel to all pairs of train vs. inducing points, and $\mathbf{K}_{tt} = \mathbf{K}(\mathbf{X}_t, \mathbf{X}_t) \in \mathbb{R}^{P_t \times P_t}$ is the kernel matrix formed by applying the kernel to all pairs of train points.

We assume our approximate posterior factorises as follows [1615],

$$q(\mathbf{f}) = q(\mathbf{f}_i, \mathbf{f}_t) = p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i), \quad q(\mathbf{f}_i) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{ii}). \quad (26.38)$$

In other words, we only model an approximate posterior $q(\mathbf{f}_i)$ on the inducing points, and use the prior conditional $p(\mathbf{f}_t | \mathbf{f}_i)$ to give the implied distribution over train points. Note that, unlike in the previous chapter, our approximate posterior is no longer over the train latents alone, but also the inducing latents $\mathbf{f}_i$. The implied approximate posterior over just the train latents $\mathbf{f}_t$ is obtained by marginalising out the inducing points,

$$q(\mathbf{f}_t) = \int p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i) d\mathbf{f}_i. \quad (26.39)$$

Since this is a Gaussian integral, it has a closed form solution, which is essentially identical to Eq. (26.30), i.e.

$$q(\mathbf{f}_t) = \mathcal{N}(\mathbf{f}_t; \mathbf{K}_{ti}^T \mathbf{K}_{ii}^{-1} \boldsymbol{\mu}_i, \mathbf{K}_{tt} - \mathbf{K}_{ti}^T \mathbf{K}_{ii}^{-1} \mathbf{K}_{it} + \mathbf{K}_{ti}^T \mathbf{K}_{ii}^{-1} \boldsymbol{\Sigma}_{ii} \mathbf{K}_{ii}^{-1} \mathbf{K}_{it}) \quad (26.40)$$

and so the ELBO given by performing variational inference over just the training latents, as in the previous section, is

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{f}_t)} [\log p(\mathbf{y}_t | \mathbf{f}_t)] - D_{\text{KL}}(q(\mathbf{f}_t) || p(\mathbf{f}_t)) \quad (26.41)$$

with $p(\mathbf{f}_t) = \mathcal{N}(\mathbf{f}_t; \mathbf{0}, \mathbf{K}_{tt})$ the prior over the training latents.

However, there is a problem with this ELBO. The KL divergence involves inverting the matrix $\mathbf{K}_{tt} \in \mathbb{R}^{P_t \times P_t}$, which is very expensive; we introduced the inducing point scheme to avoid such operations!

Titsias [1615] proposed an alternative ELBO, by directly performing variational inference over the joint distributions $q(\mathbf{f}_i, \mathbf{f}_t)$ and $p(\mathbf{f}_i, \mathbf{f}_t)$. This results in,

$$\text{ELBO}_{\text{Titsias}} = \mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}_t | \mathbf{f}_t)] - D_{\text{KL}}(q(\mathbf{f}_i, \mathbf{f}_t) || p(\mathbf{f}_i, \mathbf{f}_t)) \quad (26.42)$$

$$= \mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}_t | \mathbf{f}_t)] - D_{\text{KL}}(q(\mathbf{f}_i) || p(\mathbf{f}_i)) \quad (26.43)$$

where the second equality holds since

$$D_{\text{KL}}(q(\mathbf{f}_i, \mathbf{f}_t) || p(\mathbf{f}_i, \mathbf{f}_t)) = D_{\text{KL}}(p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i) || p(\mathbf{f}_t | \mathbf{f}_i) p(\mathbf{f}_i)) \quad (26.44)$$

$$= \mathbb{E}_{q(\mathbf{f}_i, \mathbf{f}_t)} \left[\log \frac{p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i)}{p(\mathbf{f}_t | \mathbf{f}_i) p(\mathbf{f}_i)} \right] \quad (26.45)$$

$$= \mathbb{E}_{q(\mathbf{f}_i, \mathbf{f}_t)} \left[\log \frac{q(\mathbf{f}_i)}{p(\mathbf{f}_i)} \right] \quad (26.46)$$

$$= \int \int p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i) \log \frac{q(\mathbf{f}_i)}{p(\mathbf{f}_i)} d\mathbf{f}_t d\mathbf{f}_i \quad (26.47)$$

$$= \int q(\mathbf{f}_i) \log \frac{q(\mathbf{f}_i)}{p(\mathbf{f}_i)} \left(\int p(\mathbf{f}_t | \mathbf{f}_i) d\mathbf{f}_t \right) d\mathbf{f}_i \quad (26.48)$$

$$= \int q(\mathbf{f}_i) \log \frac{q(\mathbf{f}_i)}{p(\mathbf{f}_i)} d\mathbf{f}_i \quad (26.49)$$

$$= D_{\text{KL}}(q(\mathbf{f}_i) || p(\mathbf{f}_i)). \quad (26.50)$$

Critically, when we perform variational inference over the joint process, our KL divergence simplifies to only include the inducing point distributions, which have covariance matrices of size P_i , resulting in $\mathcal{O}(P_i^3)$ computations which are much more tractable than $\mathcal{O}(P_t^3)$.

Furthermore, as shown in Hensman et al. [737], an i.i.d. likelihood function means the expected log-likelihood term decomposes into a sum over training datapoints,

$$\mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}_t | \mathbf{f}_t)] = \sum_{p=1}^{P_t} \mathbb{E}_{q(\mathbf{f})} [\log p(y_p^t | f_p^t)] \quad (26.51)$$

$$= \sum_{p=1}^{P_t} \mathbb{E}_{p(f_p^t | \mathbf{f}_i) q(\mathbf{f}_i)} [\log p(y_p^t | f_p^t)], \quad (26.52)$$

where f_p^t denotes the p^{th} training latent. In the second equality we point out that each expectation term does not rely on the joint $q(\mathbf{f}) = p(\mathbf{f}_t | \mathbf{f}_i) q(\mathbf{f}_i)$ over all training points but only the marginal $p(f_p^t | \mathbf{f}_i) q(\mathbf{f}_i)$ over a single training point (which can be obtained using a similar marginalisation process to Eq. (26.39)). In other words, when using inducing points, the training latents are conditionally independent given the inducing latents, unlike in the non-sparse approximate posterior in Section 26.2.2, where the training latents are conditional on all the other training latents. Crucially, this means that each term in the sum can be estimated independently of all other datapoints, and hence we can estimate the expected log-likelihood with minibatches of data. This is similar to minibatching in neural networks, which allow training on very large datasets without exceeding device memory or requiring the entire dataset for a single step of gradient descent.

In summary, even when using the same form of approximate posterior (Eq. 26.38), we have a choice on which objective function (i.e. which ELBO) to optimise it with. Marginalising out the inducing points first and minimising the KL divergence $D_{\text{KL}}(q(\mathbf{f}_t) || p(\mathbf{f}_t | \mathbf{y}_t))$ between the approximate posterior and the true posterior over the training latents can be rewritten as an ELBO where we needed to compute the KL divergence $D_{\text{KL}}(q(\mathbf{f}_t) || p(\mathbf{f}_t))$ between the marginalised approximate posterior and the prior over training latents, which is an $\mathcal{O}(P_t^3)$ operation. The approach introduced by Titsias [1615] instead suggests to minimise the KL divergence $D_{\text{KL}}(q(\mathbf{f}_t, \mathbf{f}_i) || p(\mathbf{f}_i, \mathbf{f}_t | \mathbf{y}_t))$ between the approximate posterior and the true posterior over the joint inducing and training latents (and marginalise out the inducing points later), which results in a more tractable ELBO because the KL divergence between the approximate posterior and prior joint simplifies as $D_{\text{KL}}(q(\mathbf{f}_i, \mathbf{f}_t) || p(\mathbf{f}_i, \mathbf{f}_t)) = D_{\text{KL}}(q(\mathbf{f}_i) || p(\mathbf{f}_i))$ which is a much cheaper $\mathcal{O}(P_i^3)$ operation. The expected log-likelihood term is the same in both approaches, and can be estimated using minibatching. Hence the only difference is the KL divergence used in the ELBO. For completeness, we note that, by the data processing inequality, $D_{\text{KL}}(q(\mathbf{f}_t) || p(\mathbf{f}_t)) \leq D_{\text{KL}}(q(\mathbf{f}_i, \mathbf{f}_t) || p(\mathbf{f}_i, \mathbf{f}_t)) = D_{\text{KL}}(q(\mathbf{f}_i) || p(\mathbf{f}_i))$, and so the bound introduced in Eq. 26.42 [1615] is potentially a looser bound on the model evidence / marginal likelihood than the bound in Eq. 26.41. However, the literature shows this bound works well in practice [737, 738, 1615].

26.2.4 Deep Gaussian Processes

Deep Gaussian processes [DGPs; 361] were introduced in detail in Chapter 25; we recap the essentials here. The Deep Kernel Machines (DKMs) and Deep Kernel Processes (DKPs) we introduce later will be based on these. A DGP maps inputs $\mathbf{X} \in \mathbb{R}^{P \times N_0}$ to outputs $\mathbf{Y} \in \mathbb{R}^{P \times N_{L+1}}$ using L intermediate layers of features $\mathbf{F}^\ell \in \mathbb{R}^{P \times N_\ell}$.

Here P is the number of input points, and N_ℓ is the number of features at layer ℓ , so that N_0 is the number of input features, and N_{L+1} is the number of output features. Writing $\mathbf{F}^\ell$ and $\mathbf{Y}$ as stacks of vectors,

$$\mathbf{F}^\ell = (\mathbf{f}_1^\ell \quad \mathbf{f}_2^\ell \quad \cdots \quad \mathbf{f}_{N_\ell}^\ell) \in \mathbb{R}^{P \times N_\ell} \quad (26.53a)$$

$$\mathbf{Y} = (\mathbf{y}_1 \quad \mathbf{y}_2 \quad \cdots \quad \mathbf{y}_{N_{L+1}}) \in \mathbb{R}^{P \times N_{L+1}}, \quad (26.53b)$$

where $\mathbf{f}_\lambda^\ell \in \mathbb{R}^P$ gives the values of a particular feature λ at layer ℓ for all P input points and $\mathbf{y}_\lambda \in \mathbb{R}^P$ gives the value of a particular output λ for all P input points. The features $\mathbf{F}^1, \dots, \mathbf{F}^L$ have i.i.d. Gaussian process priors whose covariance depends on the previous layer's features,

$$p(\mathbf{F}^\ell | \mathbf{F}^{\ell-1}) = \prod_{\lambda=1}^{N_\ell} \mathcal{N}(\mathbf{f}_\lambda^\ell; \mathbf{0}, \mathbf{K}(\mathbf{F}^{\ell-1})). \quad (26.54)$$

For regression the outputs are also sampled from a Gaussian process,

$$p(\mathbf{Y} | \mathbf{F}^L) = \prod_{\lambda=1}^{N_{L+1}} \mathcal{N}(\mathbf{y}_\lambda; \mathbf{0}, \mathbf{K}(\mathbf{F}^L) + \sigma^2 \mathbf{I}). \quad (26.55)$$

Other likelihoods could be used. For example, Milsom et al. [1179] uses a categorical likelihood with softmax probabilities, as in Eq. (26.19).

26.3 Deep Kernel Processes

The DGP defined successively alternates between (a) constructing a kernel using the features from the previous layer, and (b) sampling new features from that kernel. Remarkably, it turns out that for certain kernel functions, $\mathbf{K}(\cdot)$, we are able to reframe the DGP as a process over *kernel matrices* (a.k.a. gram matrices). We call this a Deep Kernel Process (DKP) [16, 1254]. In the intermediate layers of a DKP, there are no features — only the covariances between data points matter.

Since DKPs are equivalent to DGPs, you may be wondering why we would worry at all about replacing the features with kernel/Gram matrices. One big motivation is the large number of symmetries involved when modelling a posterior over features. The main culprit is the large number of permutation symmetries — if we swap the order of the features in a NN (or a DGP), then the model's output distribution is identical, and thus the posterior over features ought to account for this by assigning equal probability density to both feature permutations. The DKP completely sidesteps this issue by modelling the posterior over Gram matrices, which are invariant to feature permutations. To see why, let $\mathbf{G}$ be the Gram matrix formed from the features $\mathbf{F}$, then

$$\mathbf{G} = \mathbf{F}\mathbf{F}^T. \quad (26.56)$$

We can permute, or even rotate, and reflect the features, and the Gram matrix remains unchanged. This is because these operations are equivalent to multiplying the features by an orthogonal matrix $\mathbf{O}$, i.e. $\mathbf{F}' = \mathbf{F}\mathbf{O}$, where $\mathbf{O}^T \mathbf{O} = \mathbf{O}\mathbf{O}^T = \mathbf{I}$,

$$\mathbf{F}'(\mathbf{F}')^T = \mathbf{F}\mathbf{O}\mathbf{O}^T\mathbf{F}^T = \mathbf{F}\mathbf{F}^T = \mathbf{G}. \quad (26.57)$$

Full Bayesian inference is still intractable for the DKP (just like it is intractable for the DGP), but any approximate posterior we choose will by default account for a plethora of symmetries in our model. For context, even if we only think about removing the permutation symmetries, the kernel reframing removes $N!$ symmetries from our approximate posterior, where N is the number of features. Even for modestly sized networks, this is *enormously* large. A more detailed discussion on symmetries in Bayesian machine learning can be found in Chapter 32.

26.3.1 Constructing a Deep Kernel Process

Recall the DGP above, where at every layer ℓ , feature vectors $\mathbf{f}_\lambda^\ell \in \mathbb{R}^P$ are sampled i.i.d. from the kernel $\mathbf{K}(\mathbf{F}^{\ell-1})$,

$$p(\mathbf{F}^\ell | \mathbf{F}^{\ell-1}) = \prod_{\lambda=1}^{N_\ell} \mathcal{N}(\mathbf{f}_\lambda^\ell; \mathbf{0}, \mathbf{K}(\mathbf{F}^{\ell-1})). \quad (26.58)$$

We will concretely demonstrate that we can write the DGP as a kernel process. Define the rescaled Gram matrix for each layer $\mathbf{G}^\ell = \nu_\ell^{-1} \mathbf{F}^\ell (\mathbf{F}^\ell)^T$. The rescaling ensures $\mathbb{E}[\mathbf{G}^\ell] = \mathbf{K}(\mathbf{F}^{\ell-1})$. Suppose we set the kernel to be the RBF kernel, i.e. $\mathbf{K}(\mathbf{F})_{ij} = \exp(-(2\sigma^2)^{-1} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2)$. Remarkably, we are able to compute the kernel at each layer in the DGP using only the Gram matrix $\mathbf{G}^\ell$, since we can write the distance between the datapoints i and j in terms of $\mathbf{G}^\ell$,

$$R_{ij}^\ell := \nu_\ell^{-1} \|\mathbf{f}_i^\ell - \mathbf{f}_j^\ell\|_2^2 = \nu_\ell^{-1} \sum_{\lambda=1}^{\nu_\ell} (F_{i\lambda}^\ell - F_{j\lambda}^\ell)^2 = \nu_\ell^{-1} \sum_{\lambda=1}^{\nu_\ell} ((F_{i\lambda}^\ell)^2 + (F_{j\lambda}^\ell)^2 - 2F_{i\lambda}^\ell F_{j\lambda}^\ell) \quad (26.59)$$

$$= G_{ii}^\ell + G_{jj}^\ell - 2G_{ij}^\ell. \quad (26.60)$$

The DGP is therefore equivalent to,

$$p(\mathbf{F}^\ell | \mathbf{G}^{\ell-1}) = \prod_{\lambda=1}^{\nu_\ell} \mathcal{N}(\mathbf{f}_\lambda^\ell; \mathbf{0}, \mathbf{K}_\mathbf{G}(\mathbf{G}^{\ell-1})). \quad (26.61)$$

where,

$$\mathbf{K}_\mathbf{G}(\mathbf{G}^{\ell-1})_{ij} = \exp\left(-\frac{1}{2\sigma^2} (G_{ii}^{\ell-1} + G_{jj}^{\ell-1} - 2G_{ij}^{\ell-1})\right). \quad (26.62)$$

Note that the property we used above was *isotropy* of the RBF kernel, and the same idea can be applied to other isotropic kernels, such as Matern (and even the ReLU kernel when we have a large number of features). Also see the subscript in $\mathbf{K}_\mathbf{G}$ used to denote that this is a kernel formed from a Gram matrix rather than features; however it is clear from context whether we are forming the kernel using a Gram matrix or with features, so we omit the subscript henceforth.

The final step is to observe that we can sample the Gram matrix $\mathbf{G}^\ell$ directly without computing the features $\mathbf{F}^\ell$ by observing that the Gram matrix is Wishart distributed [1551, 1646]. This is true by definition, since we say a matrix $\mathbf{S} \sim \mathcal{W}(\mathbf{V}, N)$ is Wishart distributed with positive-definite scale matrix $\mathbf{V}$ and degrees of freedom N if,

$$\mathbf{x}_1, \dots, \mathbf{x}_N \sim \text{i.i.d. } \mathcal{N}(\mathbf{0}, \mathbf{V}), \quad (26.63)$$

$$\sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T = \mathbf{S} \sim \mathcal{W}(\mathbf{V}, N). \quad (26.64)$$

Care must be taken when writing the Wishart probability density function. In the case that $N < \text{rank}(\mathbf{V})$, the matrix $\mathbf{S}$ is low-rank, and one must consult the *singular* Wishart distribution [1646].

Our Gram matrix is given by $\mathbf{G}^\ell = \nu_\ell^{-1} \sum_{i=1}^{\nu_\ell} \mathbf{f}_i^\ell (\mathbf{f}_i^\ell)^T$, where $\mathbf{f}_i^\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}^{\ell-1}))$. Thus, the DGP prior for intermediate layers is equivalent to,

$$p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1}) \sim \mathcal{W}(\mathbf{G}^\ell; \mathbf{K}(\mathbf{G}^{\ell-1}) / \nu_\ell, \nu_\ell). \quad (26.65)$$

Since the distribution over the Gram matrices is Wishart, we call Eq. (26.65) a deep Wishart process [DWPs 16]. While the Wishart distribution is very natural, due to the connection with DGPs, we can imagine placing other distributions over the Gram matrices to obtain different kernel processes. For example, using the inverse Wishart distribution, gives a ‘deep inverse Wishart process’ (DIWP). The constraint is that the distribution must be over positive semi-definite Gram matrices.

For the top-layer, we need to map intermediate Gram matrices back to our data. For example, if data is Gaussian, then the following is a possible prior:

$$p(\mathbf{Y} | \mathbf{G}^L) = \prod_{\lambda=1}^{\nu_{L+1}} \mathcal{N}(\mathbf{y}_\lambda; \mathbf{0}, \mathbf{K}(\mathbf{G}^L) + \sigma^2 \mathbf{I}). \quad (26.66)$$

Note that this is analogous to the DGP top-layer prior, Eq. (26.55), but in terms of Gram matrices rather than features.

26.3.2 Bayesian Inference in a Deep Wishart Process

Similarly to how one might use Variational Inference (VI) in a DGP, Ober and Aitchison [1250] introduced a variational inference scheme for a deep Wishart process. In the literature, approximate posteriors have been

Table 26.1: (From [1250]) ELBOs and log-likelihoods for UCI datasets [566] for a five-layer network. Significantly better results (within ELBO or LL) are bolded.

Dataset	DWP ELBO	DGP ELBO	DWP LL	DGP LL
boston	-0.38 ± 0.01	-0.45 ± 0.01	-2.39 ± 0.04	-2.44 ± 0.04
concrete	-0.49 ± 0.00	-0.50 ± 0.00	-3.13 ± 0.01	-3.14 ± 0.02
energy	1.41 ± 0.00	1.39 ± 0.00	-0.70 ± 0.03	-0.70 ± 0.03
kin8nm	-0.14 ± 0.00	-0.13 ± 0.00	1.40 ± 0.00	1.38 ± 0.01
naval	3.65 ± 0.07	3.91 ± 0.10	8.21 ± 0.05	8.31 ± 0.07
power	0.03 ± 0.00	0.02 ± 0.00	-2.77 ± 0.01	-2.78 ± 0.01
protein	-1.01 ± 0.00	-1.00 ± 0.00	-2.73 ± 0.00	-2.73 ± 0.00
wine	-1.19 ± 0.00	-1.19 ± 0.00	-0.96 ± 0.01	-0.96 ± 0.01
yacht	1.65 ± 0.01	1.37 ± 0.03	-0.43 ± 0.10	-0.88 ± 0.08

proposed for both DWPs and DIWPs [16, 1250, 1254], but in this chapter we focus on the DWP, and a generalised Wishart approximate posterior.

The goal is to perform Bayesian inference over the Gram matrices $\mathbf{G}^\ell$ in a DWP,

$$p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1}) \sim \mathcal{W}(\mathbf{G}^\ell; \mathbf{K}_\mathbf{G}(\mathbf{G}^{\ell-1})/\nu_\ell, \nu_\ell). \quad (26.67)$$

Thus, we desire a flexible distribution over positive semi-definite matrices $\mathbf{G}^\ell$ to serve as our approximate posterior. This is highly unusual, and a significant departure from more usual Gaussian approximate posterior distributions we see over features. Multiple possible approximate posteriors have been proposed for the Gram matrices $\mathbf{G}$ [1250, 1250, 1254], though here we introduce the *Generalised Singular Wishart* [1254]. The idea is to parameterise $\mathbf{G}$ as,

$$\mathbf{G} = \mathbf{L} \mathbf{A} \mathbf{A}^T \mathbf{L}^T. \quad (26.68)$$

Here, $\mathbf{L}$ is a Cholesky decomposition of a positive definite scale matrix $\mathbf{V} = \mathbf{L} \mathbf{L}^T$, and $\mathbf{A}$ is a matrix of independent Gamma random variables,

$$\mathbf{A} = \begin{pmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{\nu 1} & \cdots & A_{\nu \nu} \\ \vdots & \vdots & \vdots \\ A_{P1} & \cdots & A_{P\nu} \end{pmatrix}, \quad (26.69a)$$

$$p(A_{jj}^2) = \text{Gamma}(A_{jj}^2; \alpha_{jj}, \beta_{jj}), p(A_{i>j}) = \mathcal{N}(A_{ij}; \mu_{ij}, \sigma_{ij}^2). \quad (26.69b)$$

When $\mathbf{A}$ is a tall and skinny matrix, we have that $\mathbf{A} \mathbf{A}^T$ is singular, and therefore the distribution supports positive semi-definite, but not positive definite matrices — hence ‘Generalised *Singular* Wishart’.

To actually use the Generalised Singular Wishart for VI, we need to be able to (a) sample from it and (b) evaluate its density. Sampling is straightforward; first sample $\mathbf{A}$ using Eqs. (26.69), and then use Eq. (26.68) to calculate $\mathbf{G}$. Calculating the density is very non-obvious, but [1250] showed that $\mathbf{G}$ has the following density,

$$\mathcal{GSW}(\mathbf{G} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \left(\prod_{j=1}^P \frac{1}{L_{jj}^{\min(j, \nu)}} \right) \prod_{j=1}^{\min(P, \nu)} \frac{\text{Gamma}(A_{jj}^2; \alpha_{jj}, \beta_{jj})}{A_{jj}^{P-j} L_{jj}^{P-j+1}} \prod_{i=j+1}^P \mathcal{N}(A_{ij}; \mu_{ij}, \sigma_{ij}^2). \quad (26.70)$$

By using an inducing-point version of this approximate posterior, [1250] were able to demonstrate performance improvements over similar DGP models (see Table 26.1)

26.4 Deep Kernel Machines

In this section, we introduce deep kernel machines [DKMs; 1792], a class of supervised learning models that have multiple layers of learned kernel matrices. To construct a DKM, we start with the same Wishart prior that we used for the DKP. Instead of proceeding with VI immediately however, which can become complex due to requiring

an approximate posterior over the latent Gram matrices, we first take a particular infinite width limit called the Bayesian Representation Learning Limit (BRL). The BRL turns out to be desirable because the posterior over Gram matrices becomes a Dirac-delta distribution, which means we can model the optimal Gram matrices as constant, and we end up learning them via gradient descent.

Before introducing the BRL and DKMs, we first discuss standard infinite-width neural networks, NNGPs.

26.4.1 Infinite-Width Neural Network Limits

Neural networks with infinitely many features, under Gaussian priors on the weights, are equivalent to Gaussian processes with particular kernel functions [294, 1000, 1226, 1759]. In this section, we will review the so-called “Neural Network Gaussian Process” (NNGP) limit [1000] of neural networks to aid understanding of the BRL. The infinite-width / NNGP correspondence is also briefly discussed in Chapter 31.

Consider an L -layer neural network with $N_\ell = N \cdot \nu_\ell$ hidden features at layer ℓ , and weights $\mathbf{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$. We omit biases for brevity. The preactivations and postactivations are given by,

$$F_{i\lambda}^\ell = \sum_{\mu=1}^{N_{\ell-1}} W_{\lambda\mu}^\ell H_{i\mu}^{\ell-1}, \quad H_{i\lambda}^\ell = \phi(F_{i\lambda}^\ell), \quad H_{i\lambda}^0 = X_{i\lambda}. \quad (26.71)$$

We assume zero-mean random i.i.d. initialisation of the weights, with variance σ_w^2/N_ℓ . Starting at the first layer, the central limit theorem tells us that, in the limit of infinitely many features $N_1 \rightarrow \infty$, the preactivations $\mathbf{F}^1$ will be Gaussian distributed. Lee et al. [1000] formalise this argument using induction to extend it to all layers in the network, whilst de G. Matthews et al. [386] provides an alternative argument where all layers are taken to the infinite-width limit simultaneously, rather than sequentially. We omit the details here, but the result is that the preactivations $\mathbf{f}_\lambda^\ell$ have multivariate Gaussian distribution $\mathbf{f}_\lambda^\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^\ell)$, where the covariance matrix $\mathbf{K}^\ell \in \mathbb{R}^{P \times P}$ is given by the recursion,

$$K_{ij}^\ell = \sigma_w^2 \mathbb{E}_{\mathbf{f}_\lambda^{\ell-1} \sim \mathcal{N}(\mathbf{0}, K^{\ell-1})} \left[\phi(F_{i\lambda}^{\ell-1}) \phi(F_{j\lambda}^{\ell-1}) \right]. \quad (26.72)$$

which depends on the choice of nonlinearity ϕ and the initialisation variance of the weights σ_w^2/N^ℓ . When ϕ is the ReLU nonlinearity, the recursion has a closed-form expression, the arccos kernel [294, 1000]:

$$K_{ij}^\ell = \frac{\sigma_w^2}{2\pi} \sqrt{K_{ii}^{\ell-1} K_{jj}^{\ell-1}} (\sin \theta_{ij}^{\ell-1} + (\pi - \theta_{ij}^{\ell-1}) \cos \theta_{ij}^{\ell-1}) \quad (26.73)$$

$$\theta_{ij}^{\ell-1} = \arccos \left(\frac{K_{ij}^{\ell-1}}{\sqrt{K_{ii}^{\ell-1} K_{jj}^{\ell-1}}} \right). \quad (26.74)$$

Regardless of the choice of nonlinearity, the base case is for datapoints $\mathbf{x}_i \in \mathbb{R}^{N_0}$ at the beginning of the network, with weight variance σ_w^2/N_0 , which gives the covariance matrix,

$$K_{ij}^0 = \sigma_w^2 \left(\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{N_0} \right). \quad (26.75)$$

Hence, we can perform Bayesian inference with an infinite-width neural network prior by recursively computing covariances $\mathbf{K}^\ell$, and using a Gaussian process prior with covariance matrix $\mathbf{K}^L$ to perform inference on our datapoints. This is known as the “Neural Network Gaussian Process” (NNGP) limit [1000], with the kernels referred to as NNGP kernels.

NNGPs are useful because they frame neural networks as equivalent to kernel methods, for which many theoretical properties are understood (e.g. convergence properties, generalisation bounds, etc.). However, as we will see later, the fact that they simplify to fixed kernel functions means that they do not perform representation learning, which is a key strength of deep learning, and so theoretical conclusions based on NNGPs may not transfer to practical networks. Hence, in the next section, we will modify the NNGP limit so that the resulting models (deep kernel machines) can learn representations.

26.4.2 Standard infinite-width limit

We now consider the DGP analogue of the standard NNGP infinite-width limit, that is, we will take the intermediate layers to have infinite width / infinitely-many features $\mathbf{F}^\ell \in \mathbb{R}^{P \times N_\ell}$:

$$N_\ell = N \nu_\ell \quad \text{for } \ell \in \{1, \dots, L\} \quad \text{with } N \rightarrow \infty. \quad (26.76)$$

The log-posterior (i.e. the log-joint plus a normalising constant) for the DGP written in terms of Gram matrices is,

$$\log p(\mathbf{G}^1, \dots, \mathbf{G}^L | \mathbf{X}, \mathbf{Y}) = \log p(\mathbf{Y} | \mathbf{G}^L) + \sum_{\ell=1}^L \log p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1}) + \text{const}, \quad (26.77)$$

where we emphasise that $\mathbf{G}^1, \dots, \mathbf{G}^L$ are $P \times P$ matrix arguments chosen by us to specify where we wish to evaluate the log-posterior, and so are not affected by any infinite-width limits we take. That said, the actual value of $\log p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1})$ is affected by the limit. In particular, in the infinite limit specified in Eq. (26.76), the log-prior term $\log p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1})$ (Eq. 26.65) scales with N_ℓ and therefore N , so we must divide by N to obtain a finite limit:

$$\begin{aligned} \lim_{N \rightarrow \infty} \frac{1}{N} \log p(\mathbf{G}^\ell | \mathbf{G}^{\ell-1}) &= \frac{\nu_\ell}{2} \log |\mathbf{K}(\mathbf{G}^{\ell-1})^{-1} \mathbf{G}^\ell| - \frac{\nu_\ell}{2} \text{Tr}(\mathbf{K}(\mathbf{G}^{\ell-1})^{-1} \mathbf{G}^\ell) + \lim_{N \rightarrow \infty} \frac{\alpha_\ell}{N} \\ &= -\nu_\ell \text{D}_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}^\ell) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}^{\ell-1}))) + \text{const}. \end{aligned} \quad (26.78)$$

where $\lim_{N \rightarrow \infty} \alpha_\ell / N$ is shown to converge to a constant in Yang et al. [1792]. Note that this limit has been written as a KL-divergence between two Gaussians. By contrast, the log-likelihood term $\log p(\mathbf{Y} | \mathbf{G}^L)$ is constant w.r.t. N , so will vanish if we divide by N in the limit. Putting this all together, the limiting log-posterior (divided by N to remain finite) is

$$\lim_{N \rightarrow \infty} \frac{1}{N} \log p(\mathbf{G}^1, \dots, \mathbf{G}^L | \mathbf{X}, \mathbf{Y}) = - \sum_{\ell=1}^L \nu_\ell \text{D}_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}^\ell) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}^{\ell-1}))) + \text{const}. \quad (26.79)$$

Now, as the log-posterior scales with N , the posterior itself converges to a point distribution at its global maximum $(\mathbf{G}^1)^*, \dots, (\mathbf{G}^L)^*$ (see Yang et al. [1792] for a more rigorous discussion of weak convergence),

$$\lim_{N \rightarrow \infty} p(\mathbf{G}^1, \dots, \mathbf{G}^L | \mathbf{X}, \mathbf{Y}) = \prod_{\ell=1}^L \delta(\mathbf{G}^\ell - (\mathbf{G}^\ell)^*). \quad (26.80)$$

From Eq. (26.79), it is clear that the unique global maximum is obtained when $(\mathbf{G}^\ell)^* = \mathbf{K}((\mathbf{G}^{\ell-1})^*)$, since this minimises the KL divergence. In other words, we can compute the MAP (maximum a posteriori) Gram matrices recursively by applying the kernel repeatedly:

$$\mathbf{G}^\ell = \mathbf{K}(\mathbf{G}^{\ell-1}) = \underbrace{\mathbf{K}(\mathbf{K}(\dots \mathbf{K}(\mathbf{G}^0)))}_{\ell \text{ times}}, \quad \mathbf{G}^0 = \frac{1}{N_0} \mathbf{X} \mathbf{X}^T. \quad (26.81)$$

Importantly, this means that the limiting posterior has no dependence on $\mathbf{Y}$, our training targets, and so it is not possible for representation learning to occur. This is a big problem, since the success of modern deep learning hinges on its ability to flexibly learn good representations from data.

26.4.3 The Bayesian representation learning limit and DKMs

Representation learning was lost because the likelihood term in the normalised posterior vanished in the infinite limit, whilst the prior term did not. The Bayesian representation learning limit [1792] retains representation learning by sending the number of output features to infinity, in addition to the intermediate features:

$$N_\ell = N \nu_\ell \quad \text{for } \ell \in \{1, \dots, L+1\} \quad \text{with } N \rightarrow \infty. \quad (26.82)$$

Note that the only difference between this limit and Eq. (26.76) is that we include layer $L+1$ (the output layer) in our limit. To account for an infinitely wide output layer, we use the following prior (a similar one can be used for classification),

$$p(\tilde{\mathbf{Y}} | \mathbf{G}^L) = \prod_{\lambda=1}^{N_{L+1}} \mathcal{N}(\tilde{\mathbf{y}}_\lambda; \mathbf{0}, \mathbf{K}(\mathbf{G}^L) + \sigma^2 \mathbf{I}). \quad (26.83)$$

where, $\tilde{\mathbf{Y}} \in \mathbb{R}^{P \times N_{L+1}}$. Since we are taking the limit $N_{L+1} \rightarrow \infty$, one might wonder where to find data with infinitely many features. To apply the DKM to finite-width data $\mathbf{Y} \in \mathbb{R}^{P \times \nu_{L+1}}$, we let $\tilde{\mathbf{Y}}$ be formed from N copies of $\mathbf{Y}$ concatenated together, i.e.

$$\tilde{\mathbf{Y}} = (\mathbf{Y} \quad \dots \quad \mathbf{Y}). \quad (26.84)$$

Note that since we assume each output is i.i.d., but apply copies of our data labels to them, we have a case of Bayesian model mismatch, and so the uncertainty quantification of the deep kernel machine may not be optimal. However, for the purposes of investigating representation learning in deep models, this approximation is good enough. Since we assumed the output channels to be i.i.d. (Eq. 26.83), the log-likelihood now scales with N :

$$\log p(\tilde{\mathbf{Y}}|\mathbf{G}^L) = N \log p(\mathbf{Y}|\mathbf{G}^L). \quad (26.85)$$

Computing the log-posterior as before, we therefore find that the likelihood no longer vanishes in the infinite limit, thus allowing representation learning:

$$\begin{aligned} \mathcal{L}(\mathbf{G}^1, \dots, \mathbf{G}^L) &= \lim_{N \rightarrow \infty} \frac{1}{N} \log p(\mathbf{G}^1, \dots, \mathbf{G}^L | \mathbf{X}, \tilde{\mathbf{Y}}) + \text{const}, \\ &= \log p(\mathbf{Y}|\mathbf{G}^L) - \sum_{\ell=1}^L \nu_{\ell} \text{D}_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}^{\ell}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}^{\ell-1}))). \end{aligned} \quad (26.86)$$

The limiting normalised log-posterior, $\mathcal{L}(\mathbf{G}^1, \dots, \mathbf{G}^L)$, forms the “DKM objective”, so named because it is used to optimise the Gram matrices $\mathbf{G}^{\ell}$ in a deep kernel machine. The first term encourages good predictions by maximising the log-likelihood of the training data $\mathbf{Y}$, whilst the other KL-divergence terms form a regulariser that encourages $\mathbf{G}^{\ell}$ to be close to $\mathbf{K}(\mathbf{G}^{\ell-1})$. The model reduces to the standard infinite-width limit (Eq. 26.81) when no data is observed (equivalently, when $\nu_{\ell} = \infty$ for all ℓ), since the KL-divergence is minimised by setting $\mathbf{G}^{\ell} = \mathbf{K}(\mathbf{G}^{\ell-1})$. Assuming the global maximum of this expression is unique, then the posterior is a point distribution around that maximum (Eq. 26.80). However, unlike in the NNGP case, we cannot simply compute the maximum by recursively applying the kernel function to our data, since the maximum now also depends on the likelihood term. Instead, we must optimise the Gram matrices using e.g. gradient ascent. Remarkably, given a set of optimal Gram matrices $(\mathbf{G}^1)^*, \dots, (\mathbf{G}^L)^*$, Yang et al. [1792] showed that the posterior distribution over features is zero-mean multivariate Gaussian (assuming the global maximum for the Gram matrices is unique),

$$p(\mathbf{f}_{\lambda}^{\ell} | \mathbf{X}, \mathbf{Y}) = \mathcal{N}(\mathbf{f}_{\lambda}^{\ell}; \mathbf{0}, (\mathbf{G}^{\ell})^*). \quad (26.87)$$

This fact allows *sparse* DKMs with variational inference, since a Gaussian approximate posterior can capture the true posterior. We discuss sparse DKMs in the next section, which extend the inducing-point framework from the GP literature to DKMs. When applying the BRLL to neural networks instead of DGPs, the posterior is not Gaussian, and Yang et al. [1792] needed to use normalising flows to accurately approximate the posterior, which adds significant complexity, and it is also not obvious how to integrate inducing points. As such, we stick to DKMs derived from DGPs.

Once the Gram matrices have been optimised according to the DKM objective using the training data, one can perform inference on test points using a method similar to that in DGPs. We detail the full prediction procedure for the sparse DKM in the next section, and omit the details for the full-rank DKM, which are very similar to the sparse DKM case.

26.4.4 Sparse DKMs

As with all naive kernel methods, directly computing the DKM objective for all data is $\mathcal{O}(P_t^3)$, where P_t is the number of datapoints, which is typically infeasible. As a DKM is ultimately an infinite-width DGP parameterised by Gram matrices rather than features, Yang et al. [1792] developed an inducing point scheme inspired by the doubly stochastic inducing point framework for DGPs [361, 1453] to reduce the computational complexity of the DKM objective. We will first give a high level overview of the scheme, before detailing the full derivation of the sparse DKM objective, closely following the original derivation in Yang et al. [1792].

In DGPs, each layer is a GP, so we learn one set of inducing points for each layer. When using inducing points, the features at each layer are partitioned into inducing points $\mathbf{F}_i^{\ell}$ and training points $\mathbf{F}_t^{\ell}$. Likewise, the kernel matrix is partitioned into blocks corresponding to inducing and training points. In a DKM, we partition $\mathbf{G}^{\ell}$ similarly,

$$\mathbf{G}^{\ell} = \begin{pmatrix} \mathbf{G}_{ii}^{\ell} & \mathbf{G}_{it}^{\ell} \\ \mathbf{G}_{ti}^{\ell} & \mathbf{G}_{tt}^{\ell} \end{pmatrix}, \quad \mathbf{K}(\mathbf{G}^{\ell}) = \begin{pmatrix} \mathbf{K}_{ii}(\mathbf{G}_{ii}^{\ell}) & \mathbf{K}_{it}(\mathbf{G}^{\ell}) \\ \mathbf{K}_{ti}(\mathbf{G}^{\ell}) & \mathbf{K}_{tt}(\mathbf{G}_{tt}^{\ell}) \end{pmatrix}, \quad (26.88)$$

with $\mathbf{G}_{ii}^{\ell} \in \mathbb{R}^{P_i \times P_i}$, $\mathbf{G}_{ti}^{\ell} \in \mathbb{R}^{P_t \times P_i}$, and $\mathbf{G}_{tt}^{\ell} \in \mathbb{R}^{P_t \times P_t}$. At each layer ℓ , $\mathbf{G}_{ii}^{\ell}$ is learnt, whilst $\mathbf{G}_{ti}^{\ell}$ and $\mathbf{G}_{tt}^{\ell}$ are predicted using $\mathbf{G}_{ii}^{\ell}$, $\mathbf{K}(\mathbf{G}^{\ell-1})$, and properties of the assumed Gaussian approximate posteriors. This results in the sparse

DKM objective,

$$\mathcal{L}_{\text{ind}}(\mathbf{G}_{\text{ii}}^1, \dots, \mathbf{G}_{\text{ii}}^L) = \text{GP-ELBO}(\mathbf{Y}_{\text{t}}; \mathbf{G}_{\text{ii}}^L, \mathbf{G}_{\text{ti}}^L, \mathbf{G}_{\text{tt}}^L) - \sum_{\ell=1}^L \nu_{\ell} \text{D}_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}_{\text{ii}}^{\ell}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}_{\text{ii}}^{\ell-1}))) \quad (26.89)$$

where GP-ELBO $(\mathbf{Y}_{\text{t}}; \mathbf{G}_{\text{ii}}^L, \mathbf{G}_{\text{ti}}^L, \mathbf{G}_{\text{tt}}^L)$ is the ELBO for a sparse GP (i.e. Eq. 26.42) with data $\mathbf{Y}_{\text{t}}$ and kernel $\mathbf{K}(\mathbf{G}^L)$, including a likelihood term which measures the performance of the model on training data. Hence the sparse DKM objective looks very similar in structure to the full-rank DKM objective (Eq. 26.86). This scheme is efficient, with time complexity $\mathcal{O}(L(P_{\text{i}}^3 + P_{\text{i}}^2 P_{\text{t}}))$, because the KL terms in the sparse DKM objective depend only on $\mathbf{G}_{\text{ii}} \in \mathbb{R}^{P_{\text{i}} \times P_{\text{i}}}$ (not $\mathbf{G}_{\text{tt}} \in \mathbb{R}^{P_{\text{t}} \times P_{\text{t}}}$). Furthermore, obtaining predictions and computing the expected log-likelihood term in the sparse DKM objective requires only the diagonal of $\mathbf{G}_{\text{tt}}^{\ell}$. Since we are using an i.i.d. likelihood, all covariances $\mathbf{K}_{\text{tt}}$ and $\mathbf{G}_{\text{tt}}$ between train / test points in the batch do not affect the final prediction (covariances between datapoints are instead implicitly encoded via the inducing points $\mathbf{G}_{\text{ii}}$), and so we only ever need to compute the diagonal in the train / test blocks.

We now present the full derivation of the sparse DKM objective, which was originally based on the "Doubly Stochastic Variational Inference for Deep Gaussian Processes" scheme from Salimbeni and Deisenroth [1453]. The majority of this derivation is a modified reproduction of Appendix M (Inducing Point DKMs) in Yang et al. [1792], the primary modification of note being that we explicitly model an approximate posterior over the top-layer features $\tilde{\mathbf{F}}_{L+1}$, which were just treated as deterministic variational parameters in Yang et al. [1792].

To do inducing-point variational inference, we explicitly introduce top-layer features $\mathbf{F}^{L+1} \in \mathbb{R}^{P \times \nu_{L+1}}$ and compute the likelihood from those features. Formally, each feature, $\mathbf{f}_1^{L+1}, \dots, \mathbf{f}_{\nu_{L+1}}^{L+1}$ is i.i.d., conditioned on $\mathbf{F}^L$,

$$p(\mathbf{F}^{L+1} | \mathbf{F}^L) = \prod_{\lambda=1}^{\nu_{L+1}} \mathcal{N}(\mathbf{f}_{\lambda}^{L+1}; \mathbf{0}, \mathbf{K}(\mathbf{G}(\mathbf{F}^L))), \quad (26.90a)$$

$$p(\mathbf{Y} | \mathbf{F}^{L+1}) = \prod_{\lambda=1}^{\nu_{L+1}} \mathcal{N}(\mathbf{y}_{\lambda}; \mathbf{f}_{\lambda}^{L+1}, \sigma^2 \mathbf{I}), \quad (26.90b)$$

where for simplicity we give the likelihood for regression, but other likelihoods (e.g. for classification) are possible. Since we are working with the DKM which uses infinite-width outputs, we also work with the features replicated N times, $\tilde{\mathbf{F}}^{L+1} \in \mathbb{R}^{P \times N \nu_{L+1}}$, which look exactly like the features at the previous layers, but with $N_{L+1} = N \nu_{L+1}$ features, where N is the number of copies of the data we are using to approximate the infinite-width limit:

$$p(\tilde{\mathbf{F}}^{L+1} | \mathbf{F}^L) = \prod_{\lambda=1}^{N_{L+1}} \mathcal{N}(\tilde{\mathbf{f}}_{\lambda}^{L+1}; \mathbf{0}, \mathbf{K}(\mathbf{G}(\mathbf{F}^L))), \quad (26.91a)$$

$$p(\tilde{\mathbf{Y}} | \tilde{\mathbf{F}}^{L+1}) = \prod_{\lambda=1}^{N_{L+1}} \mathcal{N}(\tilde{\mathbf{y}}_{\lambda}; \tilde{\mathbf{f}}_{\lambda}^{L+1}, \sigma^2 \mathbf{I}), \quad (26.91b)$$

Further, we take the total number of points, P , to be made up of P_{i} inducing points and P_{t} test/train points, so that $P = P_{\text{i}} + P_{\text{t}}$. Thus, we can separate all features, $\mathbf{F}^{\ell} \in \mathbb{R}^{P \times N_{\ell}}$, into the inducing features, $\mathbf{F}_{\text{i}}^{\ell} \in \mathbb{R}^{P_{\text{i}} \times N_{\ell}}$, and the test/train features, $\mathbf{F}_{\text{t}}^{\ell} \in \mathbb{R}^{P_{\text{t}} \times N_{\ell}}$. Likewise, we separate the inputs, $\mathbf{X}$, and outputs, $\mathbf{Y}$, into (potentially trained) inducing inputs, $\mathbf{X}_{\text{i}}$, and trained inducing outputs, $\mathbf{Y}_{\text{i}}$, and the real test/training inputs, $\mathbf{X}_{\text{t}}$, and outputs, $\mathbf{Y}_{\text{t}}$,

$$\mathbf{F}^{\ell} = \begin{pmatrix} \mathbf{F}_{\text{i}}^{\ell} \\ \mathbf{F}_{\text{t}}^{\ell} \end{pmatrix} \quad \tilde{\mathbf{F}}^{L+1} = \begin{pmatrix} \tilde{\mathbf{F}}_{\text{i}}^{L+1} \\ \tilde{\mathbf{F}}_{\text{t}}^{L+1} \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} \mathbf{X}_{\text{i}} \\ \mathbf{X}_{\text{t}} \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} \mathbf{Y}_{\text{i}} \\ \mathbf{Y}_{\text{t}} \end{pmatrix} \quad \tilde{\mathbf{Y}} = \begin{pmatrix} \tilde{\mathbf{Y}}_{\text{i}} \\ \tilde{\mathbf{Y}}_{\text{t}} \end{pmatrix} \quad (26.92)$$

We follow the usual doubly stochastic inducing point approach for DGPs [1453]. In particular, we treat all the features at intermediate layers, $\mathbf{F}^1, \dots, \mathbf{F}^L$, and the top-layer features, $\tilde{\mathbf{F}}^{L+1}$ as latent variables. The prior and approximate posterior over $\mathbf{F}^1, \dots, \mathbf{F}^L, \tilde{\mathbf{F}}^{L+1}$ are given by,

$$q(\mathbf{F}^1, \dots, \mathbf{F}^L, \tilde{\mathbf{F}}^{L+1} | \mathbf{X}) = q(\tilde{\mathbf{F}}^{L+1} | \mathbf{F}^L) \prod_{\ell=1}^L q(\mathbf{F}^{\ell} | \mathbf{F}_{\ell-1}), \quad (26.93a)$$

$$p(\mathbf{F}^1, \dots, \mathbf{F}^L, \tilde{\mathbf{F}}^{L+1} | \mathbf{X}) = p(\tilde{\mathbf{F}}^{L+1} | \mathbf{F}^L) \prod_{\ell=1}^L p(\mathbf{F}^{\ell} | \mathbf{F}_{\ell-1}), \quad (26.93b)$$

and $\mathbf{F}^0 = \mathbf{X}$. The prior and approximate posterior at each layer factorises into a distribution over the inducing points and a distribution over the test/train points,

$$q(\mathbf{F}^\ell | \mathbf{F}^{\ell-1}) = p(\mathbf{F}_t^\ell | \mathbf{F}_i^\ell, \mathbf{F}^{\ell-1}) q(\mathbf{F}_i^\ell), \quad (26.94a)$$

$$p(\mathbf{F}^\ell | \mathbf{F}^{\ell-1}) = p(\mathbf{F}_t^\ell | \mathbf{F}_i^\ell, \mathbf{F}^{\ell-1}) p(\mathbf{F}_i^\ell | \mathbf{F}_i^{\ell-1}). \quad (26.94b)$$

Similarly, for the top-layer features,

$$q(\tilde{\mathbf{F}}^{L+1} | \mathbf{F}^L) = p(\tilde{\mathbf{F}}_t^{L+1} | \tilde{\mathbf{F}}_i^{L+1}, \mathbf{F}^L) q(\tilde{\mathbf{F}}_i^{L+1}), \quad (26.95a)$$

$$p(\tilde{\mathbf{F}}^{L+1} | \mathbf{F}^L) = p(\tilde{\mathbf{F}}_t^{L+1} | \tilde{\mathbf{F}}_i^{L+1}, \mathbf{F}^L) p(\tilde{\mathbf{F}}_i^{L+1} | \mathbf{F}_i^L) \quad (26.95b)$$

Critically, the approximate posterior for the test/train points is the conditional prior $p(\mathbf{F}_t^\ell | \mathbf{F}_i^\ell, \mathbf{F}^{\ell-1})$, which is going to lead to cancellation when we compute the ELBO. Likewise, the approximate posterior over $\tilde{\mathbf{F}}_t^{L+1}$ is the conditional prior,

$$q(\tilde{\mathbf{F}}_t^{L+1} | \tilde{\mathbf{F}}_i^{L+1}, \mathbf{F}^L) = p(\tilde{\mathbf{F}}_t^{L+1} | \tilde{\mathbf{F}}_i^{L+1}, \mathbf{F}^L). \quad (26.96)$$

Concretely, the approximate posteriors and priors over inducing points for intermediate layers are given by,

$$q(\mathbf{F}_i^\ell) = \prod_{\lambda}^{N_\ell} \mathcal{N}(\mathbf{f}_{i;\lambda}^\ell; \mathbf{0}, \mathbf{G}_{ii}^\ell), \quad (26.97a)$$

$$p(\mathbf{F}_i^\ell | \mathbf{F}_i^{\ell-1}) = \prod_{\lambda}^{N_\ell} \mathcal{N}(\mathbf{f}_{i;\lambda}^\ell; \mathbf{0}, \mathbf{K}(\mathbf{G}(\mathbf{F}_i^{\ell-1}))) \quad (26.97b)$$

and for the top-layer, we let the approximate posterior over $\mathbf{F}_i^{L+1} \in \mathbb{R}^{P_i \times \nu_{L+1}}$ learn a separate mean for each channel (we usually use the same covariance for all channels to save memory), and define the approximate posterior over the infinite-copies $\tilde{\mathbf{F}}_i^{L+1} \in \mathbb{R}^{P_i \times N_{L+1}}$ by "copying" the approximate posterior for each concrete channel N times, whilst the prior assumes that the infinite copies are i.i.d. (again, this is a slightly strange thing to do, but it allows us to use the DKM to learn representations from finite-width data, as discussed in Section 26.4.3):

$$q(\mathbf{F}_i^{L+1}) = \prod_{\lambda}^{\nu_{L+1}} \mathcal{N}(\mathbf{f}_{i;\lambda}^{L+1}; \boldsymbol{\mu}_\lambda, \mathbf{A}) \quad (26.98a)$$

$$q(\tilde{\mathbf{F}}_i^{L+1}) = q(\mathbf{F}_i^{L+1})^N \quad (26.98b)$$

$$p(\tilde{\mathbf{F}}_i^{L+1} | \tilde{\mathbf{F}}_i^L) = \prod_{\lambda=1}^{N_{L+1}} \mathcal{N}(\tilde{\mathbf{f}}_{i;\lambda}^{L+1}; \mathbf{0}, \mathbf{K}(\mathbf{G}(\tilde{\mathbf{F}}_i^L))). \quad (26.98c)$$

Now we compute the ELBO (in the form of expected log-likelihood minus the KL between approximate posterior and true prior, as given in Eq. 26.42) with respect to our variational parameters

$$\text{ELBO}(\mathbf{G}_{ii}^1, \dots, \mathbf{G}_{ii}^L, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{N_{L+1}}, \mathbf{A}) = \mathbb{E}_q \left[\log p(\tilde{\mathbf{Y}}_t | \tilde{\mathbf{F}}_t^{L+1}) + \log \frac{p(\mathbf{F}^1, \dots, \mathbf{F}^L, \tilde{\mathbf{F}}_i^{L+1} | \mathbf{X})}{q(\mathbf{F}^1, \dots, \mathbf{F}^L, \tilde{\mathbf{F}}_i^{L+1} | \mathbf{X})} \right] \quad (26.99)$$

Substituting Eq. (26.93–26.96) and cancelling the $p(\mathbf{F}_t^\ell | \mathbf{F}_i^\ell, \mathbf{F}^{\ell-1})$ and $p(\tilde{\mathbf{F}}_t^{L+1} | \tilde{\mathbf{F}}_i^{L+1}, \mathbf{F}^L)$ terms in the fraction, we obtain

$$\text{ELBO}(\dots) = \mathbb{E}_q \left[\log p(\tilde{\mathbf{Y}}_t | \tilde{\mathbf{F}}_t^{L+1}) + \log \frac{p(\tilde{\mathbf{F}}_i^{L+1} | \mathbf{F}_i^L)}{q(\tilde{\mathbf{F}}_i^{L+1})} + \sum_{\ell=1}^L \log \frac{p(\mathbf{F}_i^\ell | \mathbf{F}_i^{\ell-1})}{q(\mathbf{F}_i^\ell)} \right] \quad (26.100)$$

In a DGP, $\mathbf{F}_i^\ell$ only depends on $\mathbf{F}_i^{\ell-1}$ through $\mathbf{G}_{ii}^{\ell-1}$, and the prior and approximate posterior factorise across features λ (Eq. 26.97). Thus, in the infinite limit, individual terms in the ELBO can be written,

$$\lim_{N \rightarrow \infty} \mathbb{E}_q \left[\log \frac{p(\mathbf{F}_i^\ell | \mathbf{F}_i^{\ell-1})}{q(\mathbf{F}_i^\ell)} \right] = N \nu_\ell \mathbb{E}_q \left[\log \frac{p(\mathbf{f}_{i;\lambda}^\ell | \mathbf{G}_{ii}^{\ell-1})}{q(\mathbf{f}_{i;\lambda}^\ell)} \right] \quad (26.101)$$

$$= -N \nu_\ell \text{D}_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}_{ii}^\ell) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}_{ii}^{\ell-1}))), \quad (26.102)$$

where the final equality arises when we notice that the expectation can be written as a negative KL-divergence. Similarly, for the top-layer features, recalling Eq. 26.98, we have

$$\lim_{N \rightarrow \infty} \mathbb{E}_q \left[\log \frac{p(\tilde{\mathbf{F}}_i^{L+1} | \mathbf{F}_i^L)}{q(\tilde{\mathbf{F}}_i^{L+1})} \right] = N \sum_{\lambda=1}^{\nu_{L+1}} \mathbb{E}_q \left[\log \frac{p(\mathbf{f}_{i;\lambda}^{L+1} | \mathbf{G}_{ii}^L)}{q(\mathbf{f}_{i;\lambda}^{L+1})} \right] \quad (26.103)$$

$$= -N \sum_{\lambda=1}^{\nu_{L+1}} D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_\lambda, \mathbf{A}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}_{ii}^L))). \quad (26.104)$$

Note that for the top layer we have a sum of ν_ℓ terms rather than just a factor of ν_ℓ as in the intermediate layer terms, since we have a separate μ_λ in the approximate posterior for each feature λ . Finally, we note that since our data labels $\tilde{\mathbf{Y}}_t$ are replicated N times, and the top-layer features $\tilde{\mathbf{F}}_t^{L+1}$ are also replicated N times in the same way, the expected log-likelihood term can be written as

$$\mathbb{E}_q [\log p(\tilde{\mathbf{Y}}_t | \tilde{\mathbf{F}}_t^{L+1})] = N \mathbb{E}_q [\log p(\mathbf{Y}_t | \mathbf{F}_t^{L+1})] \quad (26.105)$$

The inducing DKM objective, $\mathcal{L}_{\text{ind}}$, which is the ELBO divided by N to ensure that it remains finite in the infinite limit, can therefore be written as

$$\begin{aligned} \mathcal{L}_{\text{ind}}(\mathbf{F}_i^{L+1}, \mathbf{G}_{ii}^1, \dots, \mathbf{G}_{ii}^L) \\ &= \lim_{N \rightarrow \infty} \frac{1}{N} \text{ELBO}(\mathbf{F}_i^{L+1}, \mathbf{G}_{ii}^1, \dots, \mathbf{G}_{ii}^L) \\ &= \underbrace{\mathbb{E}_q [\log p(\mathbf{Y}_t | \mathbf{F}_t^{L+1})] - \sum_{\lambda=1}^{N_{L+1}} D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_\lambda, \mathbf{A}_\lambda) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}_{ii}^L)))}_{\text{Sparse GP ELBO}} \\ &\quad - \underbrace{\sum_{\ell=1}^L \nu_\ell D_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{G}_{ii}^\ell) \parallel \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}_{ii}^{\ell-1})))}_{\text{DKM Regularisation Terms}}. \end{aligned} \quad (26.106)$$

Note that this has almost exactly the same form as the standard DKM objective for DGPs (Eq. 26.89). In particular, the first term is a “performance” term that here depends on the quality of the predictions given the inducing points, and the second and third terms are a chain of KL-divergences, with the only difference that these KL-divergences apply only to the inducing points, and we have to deal with the top-layer separately due to the approximate posterior.

Now that we have a simple form for the ELBO, we need to compute the expected likelihood, $\mathbb{E}_q [\log p(\mathbf{Y}_t | \mathbf{F}_t^{L+1})]$. Since the top layer is a sparse GP, computing the expected log-likelihood term in the sparse DKM objective is similar to Section 26.2.3. For Gaussian likelihoods, we can compute the expected log-likelihood without a Monte-Carlo estimate, along with the concrete formula used for the Categorical likelihood. This relies on us having the last full Gram matrix $\mathbf{G}^L$, which depends on previous layers, and therefore requires us to compute the full Gram matrices, including test/train point blocks, conditioned on the optimised inducing Gram matrices $\mathbf{G}_{ii}^\ell$ at every layer. Once we have the last full Gram matrix $\mathbf{G}^L$, we can compute the expected log-likelihood as in a sparse GP.

We start by recapping the definition of the full Gram matrix,

$$\mathbf{G}^\ell = \begin{pmatrix} \mathbf{G}_{ii}^\ell & \mathbf{G}_{it}^\ell \\ \mathbf{G}_{ti}^\ell & \mathbf{G}_{tt}^\ell \end{pmatrix} \quad (26.107)$$

for both inducing points (labelled ‘i’) and test/training points (labelled ‘t’). For clarity, we have $\mathbf{G}^\ell \in \mathbb{R}^{P \times P}$, $\mathbf{G}_{ii}^\ell \in \mathbb{R}^{P_i \times P_i}$, $\mathbf{G}_{ti}^\ell \in \mathbb{R}^{P_t \times P_i}$, $\mathbf{G}_{tt}^\ell \in \mathbb{R}^{P_t \times P_t}$, where P_i is the number of inducing points, P_t is the number of train/test points and $P = P_i + P_t$ is the total number of inducing and train/test points.

The conditional distribution over $\mathbf{F}_t^\ell$ given $\mathbf{F}_i^\ell$ is,

$$p(\mathbf{F}_t^\ell | \mathbf{F}_i^\ell, \mathbf{G}_{\ell-1}) = \prod_{\lambda=1}^{\nu_\ell} \mathcal{N}(\mathbf{f}_{t;\lambda}^\ell; \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{f}_{i;\lambda}^\ell, \mathbf{K}_{tt-i}) \quad (26.108)$$

where $\mathbf{f}_{t;\lambda}^\ell$ is the λ th feature for all train/test points, $\mathbf{f}_{i;\lambda}^\ell$ is the λ th feature for all inducing points, and

$$\begin{pmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ti}^T \\ \mathbf{K}_{ti} & \mathbf{K}_{tt} \end{pmatrix} = \mathbf{K} \left(\frac{1}{N_{\ell-1}} \mathbf{F}^{\ell-1} (\mathbf{F}^{\ell-1})^T \right) = \mathbf{K}(\mathbf{G}_{\ell-1}) \quad (26.109)$$

$$\mathbf{K}_{tt-i} = \mathbf{K}_{tt} - \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti}^T. \quad (26.110)$$

Algorithm 27: DKM prediction

Inputs : parameters $\{\nu_\ell\}_{\ell=1}^L$; optimised Gram matrices $\{\mathbf{G}_{ii}^\ell\}_{\ell=1}^L$; inducing and train/test inputs $\mathbf{X}_i \in \mathbb{R}^{P_i \times \nu_0}$, $\mathbf{X}_t \in \mathbb{R}^{P_t \times \nu_0}$; inducing output means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{N_{L+1}} \in \mathbb{R}^{P_i}$ and covariance $\mathbf{A} \in \mathbb{R}^{P_i \times P_i}$

Outputs: predicted outputs $\mathbf{F}_t^{L+1}$

// Initialise full Gram matrix

$$\begin{pmatrix} \mathbf{G}_{ii}^0 & \mathbf{G}_{ti}^{0:T} \\ \mathbf{G}_{ti}^0 & \mathbf{G}_{tt}^0 \end{pmatrix} = \frac{1}{\nu_0} \begin{pmatrix} \mathbf{X}_i \mathbf{X}_i^T & \mathbf{X}_i \mathbf{X}_t^T \\ \mathbf{X}_t \mathbf{X}_i^T & \mathbf{X}_t \mathbf{X}_t^T \end{pmatrix}$$

// Propagate full Gram matrix

for $\ell = 1$ to L do

$$\begin{aligned} & \begin{pmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ti}^T \\ \mathbf{K}_{ti} & \mathbf{K}_{tt} \end{pmatrix} = \mathbf{K} \left(\begin{pmatrix} \mathbf{G}_{ii}^{\ell-1} & (\mathbf{G}_{ti}^{\ell-1})^T \\ \mathbf{G}_{ti}^{\ell-1} & \mathbf{G}_{tt}^{\ell-1} \end{pmatrix} \right) \\ & \mathbf{K}_{tt:i} \leftarrow \mathbf{K}_{tt} - \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti}^T \\ & \mathbf{G}_{ti}^\ell \leftarrow \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{G}_{ii}^\ell \\ & \mathbf{G}_{tt}^\ell \leftarrow \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{G}_{ii}^\ell \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti}^T + \mathbf{K}_{tt:i} \end{aligned}$$

// Final prediction using Gaussian process expressions

$$\begin{pmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ti}^T \\ \mathbf{K}_{ti} & \mathbf{K}_{tt} \end{pmatrix} = \mathbf{K} \left(\begin{pmatrix} \mathbf{G}_{ii}^L & (\mathbf{G}_{ti}^L)^T \\ \mathbf{G}_{ti}^L & \mathbf{G}_{tt}^L \end{pmatrix} \right)$$

$$\mathbf{f}_{t;\lambda}^{L+1} \sim \mathcal{N}(\mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \boldsymbol{\mu}_\lambda, \mathbf{K}_{tt} - \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti} + \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{A} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti})$$

In the infinite limit, the intermediate layer Gram matrices become deterministic via the law of large numbers:

$$\lim_{N \rightarrow \infty} \mathbf{G}^\ell = \lim_{N \rightarrow \infty} \frac{1}{N_\ell} \mathbf{F}^\ell (\mathbf{F}^\ell)^T = \lim_{N \rightarrow \infty} \frac{1}{N_\ell} \sum_{\lambda=1}^{N_\ell} \mathbf{f}_\lambda^\ell (\mathbf{f}_\lambda^\ell)^T = \mathbb{E}_q [\mathbf{f}_\lambda^\ell (\mathbf{f}_\lambda^\ell)^T] \quad (26.111)$$

(where the third equality follows since we are averaging an infinite number of i.i.d. terms, and q is defined as in Eq. 26.93 and Eq. 26.97) and as such $\mathbf{G}_{it}$ and $\mathbf{G}_{tt}$ become deterministic and equal to their expected values. Using Eq. (26.108), we can write,

$$\mathbf{F}_t^\ell = \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{F}_i^\ell + \mathbf{K}_{tt:i}^{1/2} \mathbf{Z}_i. \quad (26.112)$$

where $\mathbf{Z}_i$ is a matrix with i.i.d. standard Gaussian elements. Hence, $\mathbf{G}_{it}$ and $\mathbf{G}_{tt}$ can be written as

$$\mathbf{G}_{ti}^\ell = \frac{1}{N_\ell} \mathbb{E} [\mathbf{F}_t^\ell (\mathbf{F}_i^\ell)^T] \quad (26.113)$$

$$= \frac{1}{N_\ell} \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbb{E} [\mathbf{F}_i^\ell (\mathbf{F}_i^\ell)^T] \quad (26.114)$$

$$= \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{G}_{ii}^\ell \quad (26.115)$$

and

$$\mathbf{G}_{tt}^\ell = \frac{1}{N_\ell} \mathbb{E} [\mathbf{F}_t^\ell (\mathbf{F}_t^\ell)^T] \quad (26.116)$$

$$= \frac{1}{N_\ell} \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbb{E} [\mathbf{F}_i^\ell (\mathbf{F}_i^\ell)^T] \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti}^T + \frac{1}{N_\ell} \mathbf{K}_{tt:i}^{1/2} \mathbb{E} [\mathbf{Z}_i \mathbf{Z}_i^T] \mathbf{K}_{tt:i}^{1/2} \quad (26.117)$$

$$= \mathbf{K}_{ti} \mathbf{K}_{ii}^{-1} \mathbf{G}_{ii} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ti}^T + \mathbf{K}_{tt:i} \quad (26.118)$$

For the full prediction algorithm, see Alg. 27.

Chapter 27

Meta-learning deep kernel Gaussian processes

Although the deep kernel learning method introduced in Chapter 24 regularises deep kernels with type-II maximum likelihood estimation, it still tends to overfit to small datasets [1253]. This chapter introduces *Adaptive Deep Kernel Fitting with Implicit Function Theorem* (ADKF-IFT) [275], a general framework for learning deep kernel Gaussian processes on low-data tasks by interpolating between meta-learning and conventional deep kernel learning. ADKF-IFT employs a bilevel optimisation objective where generally useful feature representations are meta-learned across tasks, such that task-specific Gaussian process models estimated on top of such features achieve the lowest possible predictive loss on average. The resulting nested optimisation problem is solved using the implicit function theorem. The ADKF-IFT framework provably contains Deep Kernel Learning and Deep Kernel Transfer as special cases. Although ADKF-IFT is a completely general method, it is especially well-suited for drug discovery problems due to their low-data nature, significantly outperforming previous state-of-the-art methods on a variety of real-world few-shot molecular property prediction tasks and out-of-domain molecular property prediction and optimisation tasks.

27.1 Introduction

Many real-world applications require machine learning algorithms to make robust predictions with well-calibrated uncertainty given very limited training data. One important example is drug discovery, where practitioners not only want models to accurately predict biochemical and physicochemical properties of molecules, but also want to use models to guide the search for novel molecules with desirable properties, leveraging techniques such as Bayesian optimisation (BO) which heavily rely on accurate uncertainty estimates [557]. Despite the meteoric rise of neural networks over the past decade, their notoriously overconfident and unreliable uncertainty estimates [1582] make them generally ineffective surrogate models for BO. Instead, most contemporary BO implementations use Gaussian processes (GPs) [1382] as surrogate models due to their analytically tractable and generally reliable uncertainty estimates, even on small datasets.

Traditionally, GPs are fit on hand-engineered features (e.g., molecular fingerprints), which can limit their predictive performance on complex, structured, high-dimensional data such as molecules where designing informative features is challenging. Naturally, a number of works have proposed to improve performance by instead fitting GPs on features learned by a deep neural network, a family of models generally called Deep Kernel GPs as introduced in Chapter 24. However, there is no clear consensus about how to train these models on low-data tasks: maximising the GP marginal likelihood [750, 1769] has been shown to overfit on small datasets [1253], while meta-learning [1298] and fully-Bayesian approaches [1253] avoid this at the cost of making strong, often unrealistic assumptions.

This chapter presents a general framework called *Adaptive Deep Kernel Fitting with Implicit Function Theorem* (ADKF-IFT) [275] for training deep kernel GPs, which is especially well-suited to small datasets. ADKF-IFT essentially trains a subset of the model parameters with a meta-learning loss, and separately adapts the remaining parameters on each task using maximum marginal likelihood estimation. In contrast to previous methods which use a single loss for all parameters, ADKF-IFT is able to utilise the implicit regularisation of meta-learning to prevent overfitting while avoiding the strong assumptions of a pure meta-learning approach which may lead to underfitting.

The structure of this chapter is as follows. Section 27.3.1 presents the general ADKF-IFT framework and its

natural formulation as a bilevel optimisation problem. Section 27.3.2 explains how the implicit function theorem (IFT) can be used to efficiently solve the bilevel optimisation problem with gradient-based methods in a few-shot learning setting. Section 27.3.3 shows how ADKF-IFT can be viewed as a *generalisation* and *unification* of previous approaches based purely on single-task learning [1769] or purely on meta-learning [1298] for training deep kernel GPs. This chapter will focus on a specific practical instantiation of ADKF-IFT wherein all feature extractor parameters are meta-learned, which has a clear interpretation and obviates the need for any Hessian approximations. Section 27.3.4 argues why this particular instantiation is well-suited to retain the best properties of previously proposed methods. Motivated by the general demand for better GP models in chemistry, Section 27.5 presents an extensive empirical evaluation of ADKF-IFT on several chemical tasks, finding that it significantly improves upon previous state-of-the-art methods.

27.2 Preliminaries

27.2.1 Meta-learning and few-shot learning

Few-shot learning [1681] refers to learning on many related tasks when each task has few labelled examples [961, 1178]. It requires learning algorithms to be able to adapt to unseen tasks given few labelled examples. Few-shot learning algorithms often benefit from knowledge transfer among related low-data tasks, whereas single-task learning algorithms can easily overfit or underfit in this setting. In the few-shot learning setup, one is given a set of training tasks $\mathcal{D} = \{\mathcal{T}_t\}_{t=1}^T$ (a *meta-dataset*) and some unseen test tasks $\mathcal{D}_* = \{\mathcal{T}_*\}$. Each task $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N_{\mathcal{T}}}$ is a set of points in the domain $\mathcal{X}$ (e.g., space of molecules) with corresponding labels (continuous, categorical, etc.), and is partitioned into a *support set* $\mathcal{S}_{\mathcal{T}} \subseteq \mathcal{T}$ for meta-training and a *query set* $\mathcal{Q}_{\mathcal{T}} = \mathcal{T} \setminus \mathcal{S}_{\mathcal{T}}$ for meta-testing. Typically, the total number of training tasks $|\mathcal{D}|$ is large, while the size of each support set $|\mathcal{S}_{\mathcal{T}}|$ is small. Models for few-shot learning are typically trained to accurately predict unseen examples in the query set $\mathcal{Q}_{\mathcal{T}}$ given labelled examples in the support set $\mathcal{S}_{\mathcal{T}}$ for all tasks $\mathcal{T} \in \mathcal{D}$ during a *meta-training* phase, then evaluated by their prediction error on $\mathcal{Q}_{\mathcal{T}_*}$ given $\mathcal{S}_{\mathcal{T}_*}$ for unseen test tasks $\mathcal{T}_* \in \mathcal{D}_*$ during a *meta-testing* phase.

27.2.2 Deep kernel Gaussian processes

As introduced in Chapter 24, *Deep kernel* Gaussian processes are GPs whose covariance function is constructed by first using a neural network *feature extractor* f_{ϕ} with parameters $\phi \in \Phi$ to create feature representations $h = f_{\phi}(x), h' = f_{\phi}(x')$ of the input points x, x' , then feeding these feature representations into a standard *base kernel* $c_{\theta}(h, h')$ (e.g., an RBF kernel) [202, 226, 750, 1766, 1769]. The complete covariance function is therefore $k_{\psi}(x, x') := c_{\theta}(f_{\phi}(x), f_{\phi}(x'))$ with learnable parameters $\psi := (\theta, \phi)$. Deep kernel GPs can be viewed as performing Bayesian inference for the last linear layer of a neural network in the function space [709, 988, 1252, 1729].

The *Deep Kernel Learning* (DKL) [1769] method introduced in Chapter 24 is a single-task method for fitting a deep kernel to a dataset. DKL jointly fits both the feature extractor parameters ϕ and base kernel parameters θ by maximising the GP marginal likelihood on a single dataset (i.e., DKL essentially fits a neural network with a GP “head” to a dataset). It is well known that neural networks will easily overfit to small datasets [1462]. This overfitting also happens in DKL, despite the fact that it fits the neural network parameters using a type-II maximum likelihood approach: although early DKL papers suggested that the “model complexity” term (as measured by the log determinant of the kernel matrix) in the GP marginal likelihood objective would prevent this overfitting from happening [1769], recent follow-up work showed that this is not the case [1253] as a deep-kernel GP can simultaneously overfit to the training data and appear to have a low “model complexity”.

Deep Kernel Transfer (DKT) [1298] is a meta-learning method for fitting a deep kernel to a distribution of datasets. DKT jointly fits both the feature extractor parameters ϕ and base kernel parameters θ by maximising the expected GP marginal likelihood over a distribution of datasets (i.e., a meta-dataset). To mitigate the overfitting issue of DKL using meta-learning, DKT makes a very strong assumption that different tasks in the task distribution are drawn from an identical GP prior over functions. Explicitly, this means that the data generating process is assumed to have the same noise level, same amplitude, and same “characteristic lengthscale” for every task in the meta-dataset, which is a very restrictive assumption violated by most real-world problems. Taking drug discovery as an example, the drug properties in different datasets in a meta-dataset may have (1) highly varying noise levels, so modelling all tasks with the same amount of observation noise will not be realistic; (2) different output ranges and units for regression: for example, one task might have data in the range 1-20 μM , while another might have 0-100% inhibition, meaning that a single signal variance (kernel amplitude) will not model the data well; (3) different “characteristic lengthscales”: for some tasks, structurally similar molecules have very strongly correlated output labels, while for other tasks it is much weaker (i.e., there is much more variation in the labels of very

similar molecules), suggesting that the “characteristic lengthscale” will be different. Inevitably, trying to fit such a mis-specified model will result in a set of compromised base kernel parameters θ which fit all datasets fine on average but do not fit each individual dataset very well. This is the underfitting issue of DKT.

27.3 Adaptive Deep Kernel Fitting with Implicit Function Theorem

Let Θ and Φ respectively be the sets of base kernel and feature extractor parameters for a deep kernel GP. Denote the set of all parameters by $\Psi \equiv \Theta \times \Phi$.

27.3.1 The general ADKF-IFT framework with bilevel optimisation

The key idea of the general ADKF-IFT framework is to partition the parameters into two disjoint subsets $\Psi = \Psi_{\text{adapt}} \times \Psi_{\text{meta}}$, where only a subset of the parameters $\psi_{\text{adapt}} \in \Psi_{\text{adapt}}$ will be adapted to each individual task by minimising a train loss $\mathcal{L}_T$, with the remaining set of parameters $\psi_{\text{meta}} \in \Psi_{\text{meta}}$ meta-learned during a *meta-training* phase to yield the best possible validation loss $\mathcal{L}_V$ on average over many related training tasks (after ψ_{meta} is separately adapted to each of these tasks). This can be naturally formalised as the following *bilevel optimisation* problem:

$$\psi_{\text{meta}}^* = \arg \min_{\psi_{\text{meta}}} \mathbb{E}_{p(\mathcal{T})} [\mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}), \mathcal{T})], \quad (27.1)$$

$$\text{such that } \psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) = \arg \min_{\psi_{\text{adapt}}} \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \quad (27.2)$$

Equations 27.1 and 27.2 are most easily understood by separately considering the meta-learned parameters ψ_{meta} and the task-specific parameters ψ_{adapt} . For a given task $\mathcal{T}$ and an arbitrary value for the meta-learned parameters ψ_{meta} , in Equation 27.2 the task-specific parameters ψ_{adapt} are chosen to minimise the *train loss* $\mathcal{L}_T$ evaluated on the task’s support set $\mathcal{S}_{\mathcal{T}}$. That is, ψ_{adapt} is *adapted* to the support set $\mathcal{S}_{\mathcal{T}}$ of the task $\mathcal{T}$, with the aim of producing the best possible model on $\mathcal{S}_{\mathcal{T}}$ for the given value of ψ_{meta} . The result is a model with optimal task-specific parameters $\psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}})$ for the given meta-learned parameters ψ_{meta} and task $\mathcal{T}$. The remaining question is how to choose a value for the meta-learned parameters ψ_{meta} , knowing that ψ_{adapt} will be adapted separately to each task. In Equation 27.1, the optimal meta-learned parameters ψ_{meta}^* is obtained by minimising the expected *validation loss* $\mathcal{L}_V$ over a distribution of training tasks $p(\mathcal{T})$. There are two reasons for this choice as follows:

1. On any given task $\mathcal{T}$, the validation loss usually reflects the performance metric of interest on the query set $\mathcal{Q}_{\mathcal{T}}$ of $\mathcal{T}$ (e.g., the prediction error).
2. Since the same value of ψ_{meta} will be used for all tasks, it makes sense to choose a value whose *expected performance* is good across many tasks drawn from $p(\mathcal{T})$. That is, ψ_{meta} is chosen such that a GP achieves the lowest possible average validation loss on the query set $\mathcal{Q}_{\mathcal{T}}$ of a random training task $\mathcal{T} \sim p(\mathcal{T})$ after ψ_{adapt} is adapted to the task’s support set $\mathcal{S}_{\mathcal{T}}$.

In practice, ψ_{meta} would be optimised during a *meta-training* phase using a set of training tasks $\mathcal{D}$ to approximate Equation 27.1. After meta-training (i.e., at meta-test time), the joint GP posterior predictive distribution is used to make predictions for each unseen test task $\mathcal{T}_*$ with optimal parameters ψ_{meta}^* and $\psi_{\text{adapt}}^*(\psi_{\text{meta}}^*, \mathcal{S}_{\mathcal{T}_*})$:

$$p(\mathcal{Q}_{\mathcal{T}_*}^y | \mathcal{Q}_{\mathcal{T}_*}^x, \mathcal{S}_{\mathcal{T}_*}, \psi_{\text{meta}}^*, \psi_{\text{adapt}}^*(\psi_{\text{meta}}^*, \mathcal{S}_{\mathcal{T}_*})). \quad (27.3)$$

Note that the description above does not specify a particular choice of $\Psi_{\text{meta}}, \Psi_{\text{adapt}}, \mathcal{L}_T, \mathcal{L}_V$. This is intentional, as there are many reasonable choices for these quantities. Because of this, ADKF-IFT should be considered a *general framework*, with a particular choice for these being an *instantiation* of the ADKF-IFT framework. Examples of this are provided in Sections 27.3.3 and 27.3.4.

27.3.2 Efficient meta-training algorithm for ADKF-IFT

In general, optimising bilevel optimisation objectives such as Equation 27.1 is computationally complex, mainly because each evaluation of the objective requires solving a separate inner optimisation problem (Equation 27.2).

Algorithm 28: Exact hypergradient computation in ADKF-IFT

Input : training task $\mathcal{T}'$; current meta-learned parameters ψ'_{meta}
Output: hypergradient $\frac{d\mathcal{L}_V}{d\psi_{\text{meta}}}$ for $\mathcal{T}'$ evaluated at ψ'_{meta}
 solve Equation 27.2 to obtain $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$
 compute $g_1 = \left. \frac{\partial \mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}')}{\partial \psi_{\text{meta}}} \right|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$ and $g_2 = \left. \frac{\partial \mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}')}{\partial \psi_{\text{adapt}}} \right|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
 compute Hessian $H = \left. \frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{adapt}}^\top} \right|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
 solve the linear system $vH = g_2$ for v
 compute mixed partial derivatives $P = \left. \frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{meta}}^\top} \right|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
 compute hypergradient $\frac{d\mathcal{L}_V}{d\psi_{\text{meta}}} = g_1 - vP$

Although calculating the *hypergradient* (i.e., total derivative) of the validation loss $\mathcal{L}_V$ with respect to the meta-learned parameters ψ_{meta} would allow Equation 27.1 to be solved with gradient-based optimisation:

$$\frac{d\mathcal{L}_V}{d\psi_{\text{meta}}} = \frac{\partial \mathcal{L}_V}{\partial \psi_{\text{meta}}} + \frac{\partial \mathcal{L}_V}{\partial \psi_{\text{adapt}}^*} \frac{\partial \psi_{\text{adapt}}^*}{\partial \psi_{\text{meta}}}, \quad (27.4)$$

Equation 27.4 reveals that this requires calculating $\frac{\partial \psi_{\text{adapt}}^*}{\partial \psi_{\text{meta}}}$, i.e., how the optimal task-specific parameters $\psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$ change with respect to the meta-learned parameters ψ_{meta} . Calculating this naively with automatic differentiation platforms would require tracking the gradient through many iterations of the inner optimisation (Equation 27.2), which in practice requires too much memory to be feasible. Fortunately, because ψ_{adapt}^* is an optimum of the train loss $\mathcal{L}_T$, Cauchy's *Implicit Function Theorem* (IFT) provides a formula for calculating $\frac{\partial \psi_{\text{adapt}}^*}{\partial \psi_{\text{meta}}}$ for an arbitrary value of the meta-learned parameters ψ'_{meta} and a given task $\mathcal{T}'$:

$$\left. \frac{\partial \psi_{\text{adapt}}^*}{\partial \psi_{\text{meta}}} \right|_{\psi'_{\text{meta}}} = - \left(\frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{adapt}}^\top} \right)^{-1} \left. \frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{meta}}^\top} \right|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}, \quad (27.5)$$

where $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$. The only potential problem with Equation 27.5 is the computation and inversion of the Hessian matrix $\frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{adapt}}^\top}$. This computation can be done exactly if $|\Psi_{\text{adapt}}|$ is small, which is the case considered in this chapter (as will be discussed in Section 27.3.4). Otherwise, an approximation to the inverse Hessian (e.g., Neumann approximation [313, 1080]) could be used, which reduces both the memory and computational complexities to $\mathcal{O}(|\Psi|)$. Combining Equations 27.4 and 27.5, Algorithm 28 summarises a recipe for computing the hypergradient $\frac{d\mathcal{L}_V}{d\psi_{\text{meta}}}$ exactly for a single task. The meta-learned parameters ψ_{meta} can then be updated with the expected hypergradient over $p(\mathcal{T})$.

27.3.3 ADKF-IFT as a unification framework

The most common method used to train deep kernel GPs is to minimise the negative log marginal likelihood (NLML) on a single dataset (optionally with extra regularisation terms as will be discussed in Section 27.4.1). This is commonly referred to as *Deep Kernel Learning* (DKL) [1769] and introduced in Chapter 24:

$$\psi^* = \arg \min_{\psi} \text{NLML}(\psi, \mathcal{S}_{\mathcal{T}}). \quad (27.6)$$

The most notable departure from DKL is *Deep Kernel Transfer* (DKT) [1298], which instead proposes to train deep kernel GPs entirely using meta-learning, minimising the *expected* NLML over a distribution of training tasks:

$$\psi^* = \arg \min_{\psi} \mathbb{E}_{p(\mathcal{T})} [\text{NLML}(\psi, \mathcal{T})]. \quad (27.7)$$

Interestingly, both DKL and DKT can be viewed as *special cases* of the general ADKF-IFT framework. It is easy to see that choosing the partition to be $\Psi_{\text{meta}} = \emptyset$, $\Psi_{\text{adapt}} = \Psi$ and the train loss $\mathcal{L}_T$ to be the NLML in Equations 27.1 and 27.2 yields Equation 27.6: DKL is just ADKF-IFT if no parameters are meta-learned. Similarly, choosing the partition to be $\Psi_{\text{meta}} = \Psi$, $\Psi_{\text{adapt}} = \emptyset$ and the validation loss $\mathcal{L}_V$ to be the NLML in Equations 27.1 and 27.2 yields Equation 27.7: DKT is just ADKF-IFT if all parameters are meta-learned. This makes ADKF-IFT *strictly more general* than these two previous methods.

27.3.4 Highlighted instantiation of ADKF-IFT

Among the many possible variations of ADKF-IFT, the following instantiation is highlighted.

- $\Psi_{\text{meta}} = \Phi$, i.e., all feature extractor parameters ϕ are meta-learned across tasks.
- $\Psi_{\text{adapt}} = \Theta$, i.e., all base kernel parameters θ (e.g., noise variance, lengthscales, amplitude) are adapted to each task.
- The train loss $\mathcal{L}_T$ is chosen to be the negative log GP marginal likelihood evaluated on the support set $\mathcal{S}_T$, as is common practice for choosing GP base kernel parameters:

$$\begin{aligned}\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_T) &= -\log p(\mathcal{S}_T^y | \mathcal{S}_T^x, \psi_{\text{meta}}, \psi_{\text{adapt}}) \\ &= \frac{1}{2}(\mathcal{S}_T^y)^\top K_{\mathcal{S}_T}^{-1} \mathcal{S}_T^y + \frac{1}{2} \log \det(K_{\mathcal{S}_T}) + \frac{N_{\mathcal{S}_T}}{2} \log(2\pi),\end{aligned}\quad (27.8)$$

where $K_{\mathcal{S}_T} := k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{S}_T^x, \mathcal{S}_T^x) + \sigma^2 I_{N_{\mathcal{S}_T}}$.

- The validation loss $\mathcal{L}_V$ is chosen to be the negative log joint GP predictive posterior evaluated on the query set $\mathcal{Q}_T$ given the support set $\mathcal{S}_T$, also due to its common usage for making predictions with GPs:

$$\begin{aligned}\mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}) &= -\log p(\mathcal{Q}_T^y | \mathcal{Q}_T^x, \mathcal{S}_T, \psi_{\text{meta}}, \psi_{\text{adapt}}) \\ &= -\log \mathcal{N}(\mathcal{Q}_T^y | K_{\mathcal{Q}_T \mathcal{S}_T} K_{\mathcal{S}_T}^{-1} \mathcal{S}_T^y, K_{\mathcal{Q}_T} - K_{\mathcal{S}_T \mathcal{Q}_T}^\top K_{\mathcal{S}_T}^{-1} K_{\mathcal{S}_T \mathcal{Q}_T}),\end{aligned}\quad (27.9)$$

where $K_{\mathcal{Q}_T} := k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{Q}_T^x, \mathcal{Q}_T^x) + \sigma^2 I_{N_{\mathcal{Q}_T}}$ and $K_{\mathcal{S}_T \mathcal{Q}_T} := k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{S}_T^x, \mathcal{Q}_T^x)$.

There are several benefits to this choice as detailed below.

1. This particular choice of loss functions has the advantage that the prediction procedure during meta-testing as defined in Equation 27.3 exactly matches the meta-training procedure, thereby closely following the principle of *learning to learn*.
2. The partition of parameters can be intuitively understood as meta-learning a generally useful feature extractor f_ϕ such that it is possible on average to fit a low-loss GP to the feature representations extracted by f_ϕ for each individual task. This is very similar to previous transfer learning approaches.
3. Since most GP base kernels have only a handful of parameters, the Hessian in Equation 27.5 can be computed and inverted exactly during meta-training using Algorithm 28; this removes any need for Hessian approximations.
4. The inner optimisation (Equation 27.2) for ψ_{adapt} is computationally efficient, as it does not require back-propagating through the feature extractor f_ϕ .

More generally, ADKF-IFT combines DKL and DKT in a way that can potentially inherit the strengths of both methods and the weaknesses of neither. By adapting the base kernel parameters θ specifically to each task, it prevents underfitting due to varying ranges, lengthscales, or noise levels between datasets. By meta-learning the feature extractor on many datasets, it prevents overfitting as observed by Ober et al. [1253]. This advantage is both *theoretically principled* (by solving a bilevel optimisation objective using the implicit function theorem) and *empirically observable* (as will be shown in Section 27.5).

The relationship between these methods is visualised in Figure 27.1. Panel (c) shows DKL, which trains a separate deep kernel GP for each task. It is not hard to imagine that this can lead to severe overfitting for small datasets, which has been observed empirically by Ober et al. [1253]. Panel (b) shows DKT, which prevents overfitting by fitting one deep kernel GP for all tasks. However, this implicitly makes a strong assumption that all tasks come from an identical distribution over functions, including the same noise level, same amplitude, and same characteristic lengthscales, which is unlikely to hold in practice. Panel (a) shows ADKF-IFT, which allows these important parameters to be adapted, while still regularising the feature extractor with meta-learning. The choice of this specific instantiation is based on the assumption that adapting the base kernel parameters is more appropriate given the expected differences between tasks: two related tasks are more likely to have different noise levels or characteristic lengthscales than to require substantially different feature representations (as discussed in Section 27.2.2).

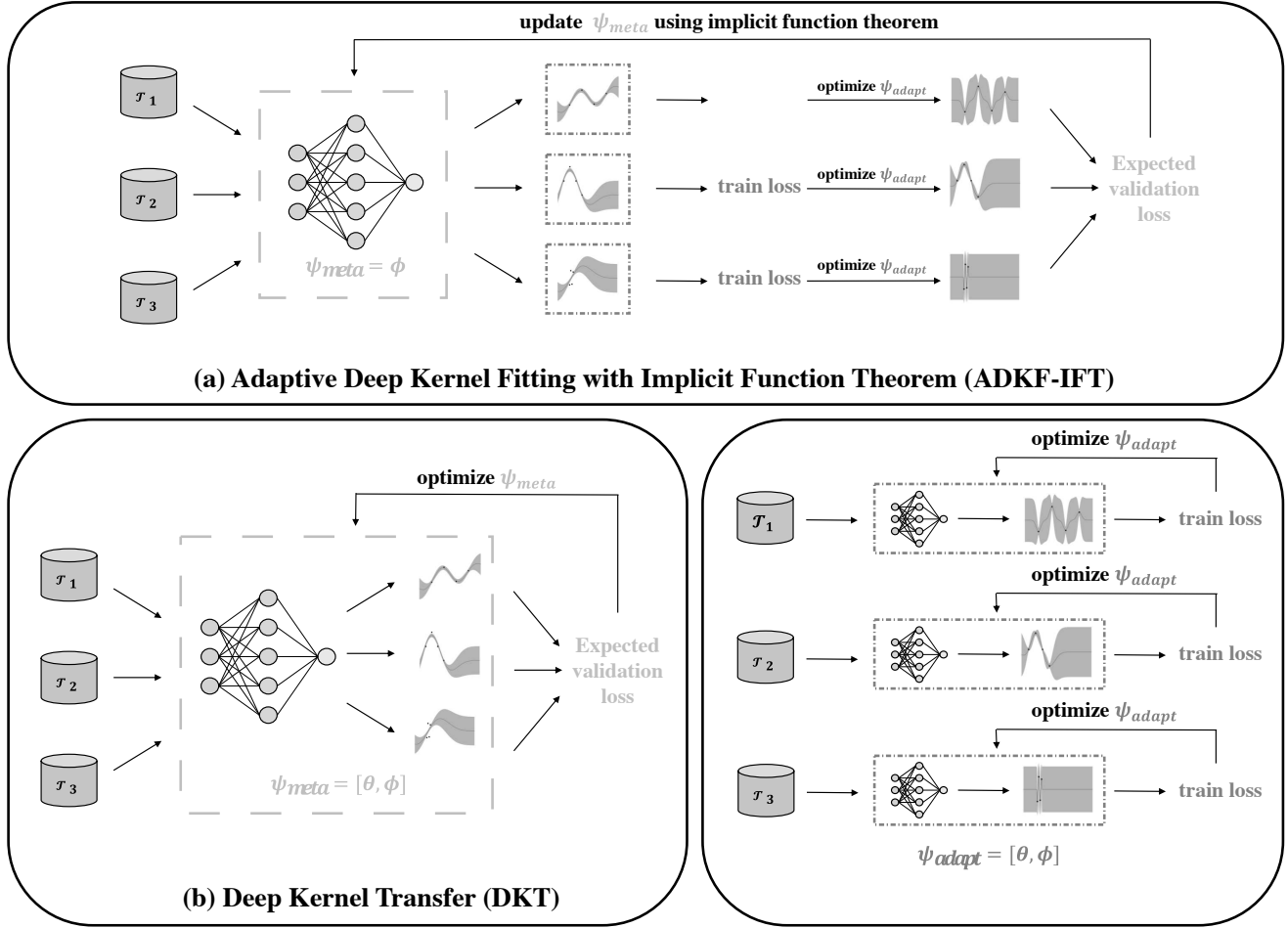


Figure 27.1: A diagram contrasting the training procedures of ADKF-IFT, DKT and DKL. (a) ADKF-IFT meta-learns the feature extractor parameters ϕ across all tasks and adapts the base kernel parameters θ to each task under a bilevel optimisation framework. (b) DKT meta-learns all parameters across all tasks. (c) DKL adapts all parameters to each task.

27.4 Connection to other methods

27.4.1 Deep kernel Gaussian processes

ADKF-IFT is part of a growing body of literature of techniques to train deep kernel GPs. As discussed in Section 27.3.3, ADKF-IFT *generalises* DKL [1769] and DKT [1298], which exclusively use single-task learning and meta-learning, respectively. Liu et al. [1061] and van Amersfoort et al. [1654] propose adding regularisation terms to the loss of DKL in order to mitigate overfitting. These works are better viewed as *complementary* to ADKF-IFT rather than *alternatives*: their respective regularisation terms could easily be added to $\mathcal{L}_T$ in Equation 27.2 to improve performance. However, the regularisation strategies in both of these papers are designed for continuous inputs only, limiting their applicability to structured data like molecules.

The preprint of Tossou et al. [1624] proposes an alternative way to meta-train adaptive deep kernel GPs called ADKL-GP, where adaptation is performed by conditioning the feature extractor on an embedding of the entire support set rather than adjusting a subset of the kernel parameters as in ADKF-IFT. In general, this method is very prone to overfitting. This is because the training objective for ADKL-GP is equivalent to the objective for DKT with an added contrastive loss, weighted by a sensitive hyperparameter γ , which can be interpreted as balancing the degree of regularisation between two extremes. For this method to be useful it would appear that γ would need to be carefully tuned to balance between these extremes. However, empirical results suggest that there is no consistent trend besides $\gamma > 0$ being slightly helpful, although the differences in performance were small. This suggests that the method may be difficult to use in practice. ADKF-IFT however has no such tunable hyperparameters, which is a significant strength. Instead, the balance between DKL and DKT is controlled by

selecting which parameters are adapted and meta-learned, which is much more interpretable and makes it easier to use in practice.

27.4.2 Meta-learning

ADKF-IFT can also be viewed as a meta-learning algorithm comparable to many previously-proposed methods [267, 284, 582, 961, 1059, 1294, 1299, 1606, 1633, 1681, 1775]. One distinguishing feature of ADKF-IFT is that it is specially designed for deep kernel GPs, whereas most methods from computer vision are designed exclusively for neural network models, which as previously stated are unsuitable when reliable uncertainty estimates are required. Furthermore, many of these algorithms such as ProtoNet [1527] are designed principally or exclusively for classification, while ADKF-IFT is suited to both regression and classification. Compared to model-agnostic frameworks like MAML [518, 1372] which meta-learn a common initialisation for adaptation of all parameters, ADKF-IFT meta-learns a subset of parameters across tasks and adapts the rest of parameters to each task.

27.4.3 Multi-Task Gaussian processes

ADKF-IFT can be considered as a method to learn multi-task GPs. The dominant approach to this problem in prior works is to learn a shared kernel for all data points across all tasks, transmitting information by explicitly modelling the covariance between data from different tasks [186, 545, 880, 1334, 1580]. The main difference between methods in this family is the exact form of the kernel [1610], which is typically assumed to have a particular structure (e.g., Kronecker product or weighted sum of simpler kernels). ADKF-IFT cannot be naturally viewed in this way because the covariance between data points from separate tasks is always zero; information is instead transmitted across tasks via a shared set of model parameters. Therefore, ADKF-IFT is a significant departure from the dominant paradigm in GP transfer learning.

27.4.4 Implicit function theorem in machine learning

The implicit function theorem employed in ADKF-IFT has been used in many previous machine learning papers in various contexts, e.g., neural architecture search [1852], hyperparameter-tuning [114, 313, 1080, 1093, 1308], and meta-learning [284, 1005, 1372].

One challenge of applying implicit function theorem is the computation of the inverse Hessian. While earlier work compute this explicitly [114, 977] as in ADKF-IFT, recent work employ various approximation techniques such as Neumann approximation [313, 1080], conjugate gradient [284, 1308, 1372, 1713], and the identity matrix [1093] to retain computational tractability.

27.5 Empirical evaluation

This section presents empirical evaluation results for the specific instantiation of ADKF-IFT from Section 27.3.4. The experiments are chosen to focus exclusively on molecular property prediction and optimisation tasks, because this application would benefit greatly from better GP models: firstly because many existing methods struggle on small datasets of size $\sim 10^2$ which are ubiquitous in chemistry, and secondly because many tasks in chemistry require high-quality uncertainty estimates. Specifically, ADKF-IFT is evaluated on a larger-scale molecular property prediction benchmark named FS-Mol [1555]. Empirical results confirm that ADKF-IFT is the best-performing method (Section 27.5.1), supporting the hypothesis from Section 27.3.4 that ADKF-IFT achieves a better balance between overfitting and underfitting than DKL and DKT. Moreover, additional Bayesian optimisation results demonstrate that the ADKF-IFT feature representation is transferable to out-of-domain (OOD) molecular property prediction and optimisation tasks (Section 27.5.2).

27.5.1 Few-shot molecular property prediction on FS-Mol

The empirical evaluation is conducted on the FS-Mol benchmark [1555], which contains a carefully constructed set of few-shot learning tasks for molecular property prediction. FS-Mol contains over 5,000 tasks with 233,786 unique compounds from ChEMBL27 [1164], split into training (4,938 tasks), validation (40 tasks), and test (157 tasks) sets. Each task is associated with a protein target. The original benchmark only considers binary classification of active/inactive compounds. Nevertheless, the regression task for the actual numeric activity target IC50 or EC50 is included in this evaluation, as it is a desired and more preferred task to do in real-world drug discovery projects.

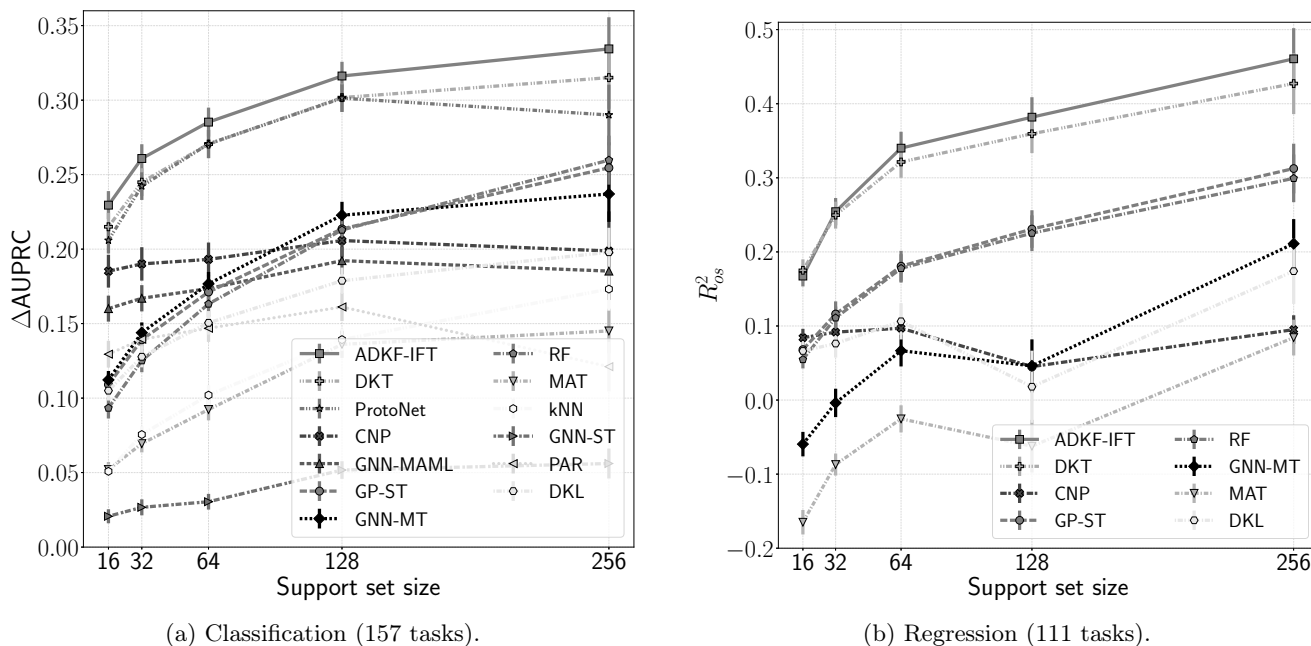


Figure 27.2: Mean performance with standard error of each compared method on all FS-Mol test tasks.

ADKF-IFT is compared with four categories of baselines: (1) *Single-task learning methods*: random forest (**RF**), k-nearest neighbours (**kNN**), single-task GP with Tanimoto kernel (**GP-ST**) [1373], single-task GNN (**GNN-ST**) [632], deep kernel learning (**DKL**) [1769]; (2) *Multi-task pretraining method*: multi-task GNN (**GNN-MT**) [331, 632]; (3) *Self-supervised pretraining method*: molecule attention transformer (**MAT**) [1153]; (4) *Meta-learning methods*: property-aware relation networks (**PAR**) [1712], prototypical network with Mahalanobis distance (**ProtoNet**) [1527], model-agnostic meta-learning (**GNN-MAML**) [518], conditional neural process (**CNP**) [582], deep kernel transfer (**DKT**) [1298].

The GNN feature extractor architecture f_ϕ used for DKL, PAR, CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet, GNN-ST, GNN-MT, and GNN-MAML in Stanley et al. [1555]. All multi-task and meta-learning methods are trained from scratch on FS-Mol training tasks. MAT is pretrained on 2 million molecules sampled from the ZINC15 dataset [1563]. The classification results for RF, kNN, GNN-ST, GNN-MT, MAT, ProtoNet, and GNN-MAML are reproduced according to Stanley et al. [1555].

The task-level metrics for binary classification and regression are ΔAUPRC (change in area under the precision-recall curve) and R^2_{oss} (predictive/out-of-sample coefficient of determination), respectively. The evaluation procedure is the same as that in Stanley et al. [1555], where the averaged performance over ten different stratified support/query random splits of every test task is reported for each compared method. This evaluation process is performed for five different support set sizes 16, 32, 64, 128, and 256. Note that unlike MoleculeNet, the support sets are generally unbalanced for the classification task in FS-Mol, which is natural as the majority of the candidate molecules are inactive in drug discovery.

Figure 27.2 shows the overall test performance of all compared methods aggregated over all tasks. Note that RF is a strong baseline method, as it is widely used in real-world drug discovery projects and has comparable performance to many pretraining methods. The results indicate that ADKF-IFT outperforms all the other compared methods at all considered support set sizes for the classification task. For the regression task, the performance gains of ADKF-IFT over the second best method, namely DKT, get larger as the support set size increases.

27.5.2 Out-of-domain molecular property prediction and optimisation

This section further demonstrates that the feature representation learned by ADKF-IFT is useful not only for in-domain molecular property prediction tasks but also for out-of-domain (OOD) molecular property prediction and optimisation tasks. For this, experiments are performed to find molecules with best desired target properties within given OOD datasets using Bayesian optimisation (BO) with a GP surrogate model operating on top of compared feature representations. These representations are evaluated on four representative molecular design tasks [579, 695, 1567, 1595] outside of FS-Mol, as summarised in Table 27.1. For the GP surrogate model,

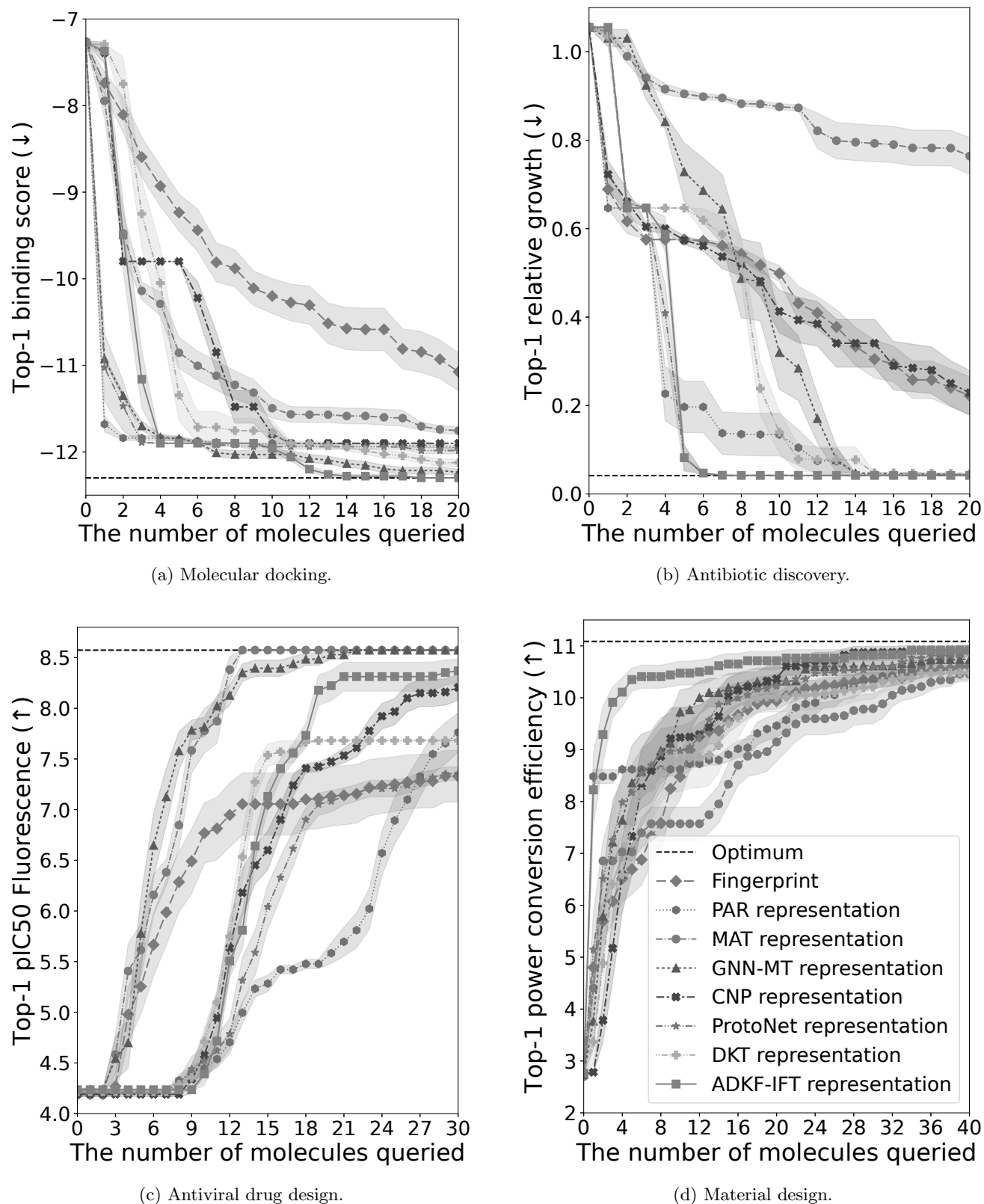


Figure 27.3: Mean top-1 target value with standard error as a function of the number of molecules queried for each compared feature representation on four out-of-domain molecular optimisation tasks.

Table 27.1: Descriptions of four out-of-domain molecular design tasks. The datasets for the molecular docking and material design tasks are sub-sampled from the much larger datasets provided in DOCKSTRING [579] and Harvard Clean Energy Project [695], respectively. The datasets for the antibiotic discovery and antiviral drug design tasks are taken from the antibiotic training set and the COVID Moonshot dataset provided in Stokes et al. [1567] and The COVID Moonshot Consortium [1595], respectively.

Molecular design task	Data source	#compounds	Target
Docking (ESR2)	DOCKSTRING training set	2,312	binding score
Antibiotic (E. coli BW25113)	Antibiotic training set	2,335	relative growth
Antiviral (SARS-CoV-2)	COVID Moonshot	1,926	pIC50 Fluorescence
Material (OPV)	Harvard Clean Energy Project	2,012	power conversion efficiency

Table 27.2: Mean predictive performance (test NLL) with standard error of a GP operating on top of each feature representations on the four out-of-domain molecular design tasks.

Feature	Out-of-domain molecular design task			
	Molecular docking	Antibiotic discovery	Antiviral drug design	Material design
Fingerprint	<u>1.138 $\pm$ 0.014</u>	1.669 $\pm$ 0.075	4.601 $\pm$ 0.086	1.091 $\pm$ 0.011
PAR	1.270 $\pm$ 0.019	2.185 $\pm$ 0.115	4.840 $\pm$ 0.086	1.283 $\pm$ 0.017
MAT	1.528 $\pm$ 0.028	2.390 $\pm$ 0.104	4.797 $\pm$ 0.088	2.198 $\pm$ 0.063
GNN-MT	1.994 $\pm$ 0.050	3.692 $\pm$ 0.225	6.399 $\pm$ 0.181	7.254 $\pm$ 0.217
CNP	1.493 $\pm$ 0.028	2.537 $\pm$ 0.162	5.005 $\pm$ 0.086	1.741 $\pm$ 0.043
ProtoNet	1.147 $\pm$ 0.013	1.615 $\pm$ 0.094	5.060 $\pm$ 0.086	1.032 $\pm$ 0.009
DKT	1.167 $\pm$ 0.012	<u>1.602 $\pm$ 0.073</u>	4.975 $\pm$ 0.092	<u>1.026 $\pm$ 0.009</u>
ADKF-IFT	1.137 $\pm$ 0.011	1.496 $\pm$ 0.043	<u>4.781 $\pm$ 0.087</u>	0.996 $\pm$ 0.007

the Tanimoto kernel is used for the fingerprint representation (with radius 2 and 2,048 bits based on count simulation). All the other compared feature representations employ Matérn-5/2 kernel without automatic relevance determination (ARD) but with a log-normal prior over the lengthscale centred at the median heuristic initialisation [585]. For BO, the expected improvement (EI) [848] acquisition function is employed with a query-batch size 1. The base kernel parameters are re-fit at the beginning of each BO iteration using all available data points. All compared feature representations are extracted using the models trained on the FS-Mol dataset from scratch in Section 27.5.1, except for the pretrained MAT representation and the fixed fingerprint representation. Each BO experiment is repeated 20 times, each time starting from 16 randomly sampled molecules from the worst ~ 700 molecules within the dataset. Figure 27.3 shows that the ADKF-IFT representation enables fastest discovery of top performing molecules for the molecular docking, antibiotic discovery, and material design tasks. For the antiviral drug design task, although the ADKF-IFT representation underperforms the MAT and GNN-MT representations, it still achieves competitive performance compared to other baselines.

Table 27.2 explicitly reports the regression predictive performance of a GP operating on top of each compared feature representation for these four out-of-domain molecular design tasks. The configuration of the GP is the same as that in the BO experiments. The test negative log likelihood (NLL) averaged over 200 support/query random splits (100 for each of the support set sizes 32 and 64) is reported. The results show that the ADKF-IFT representation has the best test NLL on the molecular docking, antibiotic discovery, and material design tasks, and ranks second on the antiviral drug design task.

27.6 Summary

This chapter presented Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), a general framework for fitting deep kernels that interpolates between meta-learning and conventional deep kernel learning. ADKF-IFT meta-learns a feature extractor across tasks such that the task-specific GP models estimated on top of the extracted feature representations can achieve the lowest possible prediction error on average. ADKF-IFT is implemented by solving a bilevel optimisation objective via implicit differentiation, which results in a unifying framework containing DKL and DKT as special cases. The empirical results demonstrated that ADKF-IFT learns generally useful feature representations, achieving state-of-the-art performance on a variety of real-world few-shot molecular property prediction tasks and on out-of-domain molecular property prediction and optimisation tasks, outperforming a wide range of single-task, multi-task, pre-training, and meta-learning methods. ADKF-IFT

could potentially be an important method to produce well-calibrated models for fully-automated high-throughput experimentation in the future.

Having said that, one potential factor that could limit the use case of ADKF-IFT is that it requires a meta-dataset of many related tasks for meta-training the feature extractor, which is not always available in practice. Furthermore, the inner optimisation procedure is a major bottleneck in ADKF-IFT since it needs to be performed until convergence in every training and inference step.

Part VI

Priors

Chapter 28

Introduction to priors

In classical Bayesian statistics, a *prior* $p(\theta)$ represents our initial beliefs about the parameters $\theta \in \mathbb{R}^P$ of a given statistical model before observing data. Prior knowledge is a cornerstone of Bayesian analysis, as it allows us to integrate existing knowledge or expert insights into our models. This is typically motivated by subjective Bayesian probability, which represents a degree of belief, but is also contrasted with objective Bayesian approaches that seek to minimise the prior's influence when no prior knowledge is available.

To illustrate classical prior elicitation, consider a highly informative prior expressing strong historical beliefs about the average heights of students. For first-grade students, we use a prior suggesting a normal distribution with mean 1.2 metres and variance 0.003. For third-grade students, the prior is a normal distribution with mean 1.4 metres and variance 0.002. Suppose we observe heights for 10 first-graders: {1.18, 1.22, 1.25, 1.20, 1.21, 1.19, 1.21, 1.24, 1.17, 1.26} metres (sample mean 1.213), and 5 third-graders: {1.37, 1.39, 1.42, 1.36, 1.40} metres (sample mean 1.388). We model the mean height θ_k for each group k , assuming heights $y_{k,i} \sim \mathcal{N}(\theta_k, 0.001)$. The

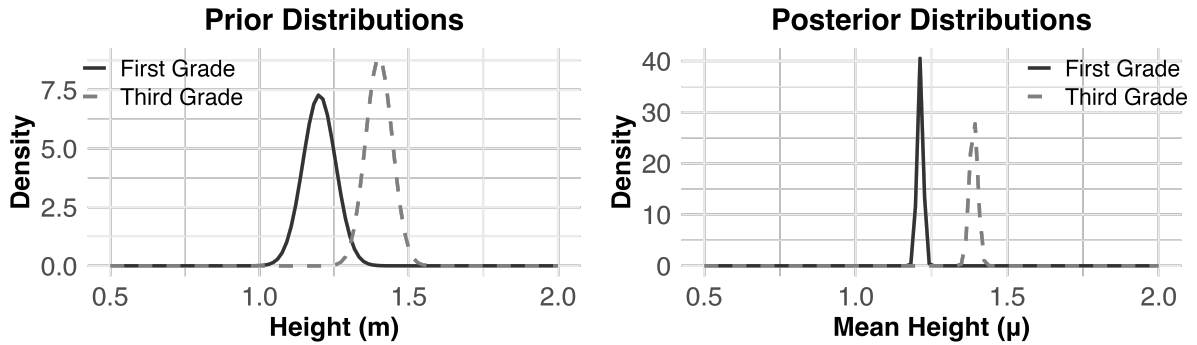


Figure 28.1: Informative priors for students' average height (solid line for first-grade: $\mathcal{N}(1.2, 0.003)$, dashed line for third-grade: $\mathcal{N}(1.4, 0.002)$) and their posteriors (right panel), showing updated beliefs after observing the data.

posterior distribution is $p(\theta_k | \mathbf{y}_k) \propto p(\theta_k)p(\mathbf{y}_k | \theta_k)$. The prior is $\mathcal{N}(\theta_k; \mu_{0,k}, \sigma_{0,k}^2)$, and the likelihood for n_k observations is:

$$p(\mathbf{y}_k | \theta_k) \propto \exp \left(-\frac{n_k}{2 \times 0.001} (\bar{y}_k - \theta_k)^2 \right).$$

Since the prior and likelihood are both Gaussian, this is a conjugate prior-likelihood pair. The posterior is also Gaussian, $\mathcal{N}(\theta_k; \mu_{n,k}, \sigma_{n,k}^2)$, with parameters given analytically by:

$$\mu_{n,k} = \frac{\frac{\mu_{0,k}}{\sigma_{0,k}^2} + \frac{n_k \bar{y}_k}{0.001}}{\frac{1}{\sigma_{0,k}^2} + \frac{n_k}{0.001}}, \quad \sigma_{n,k}^2 = \frac{1}{\frac{1}{\sigma_{0,k}^2} + \frac{n_k}{0.001}}.$$

Thus, for first-graders, the posterior is $\mathcal{N}(1.212581, 0.00009677)$; while for third-graders, it is $\mathcal{N}(1.389091, 0.0001818)$. We have significantly reduced the uncertainty after observing the data and updated our prior expectation, as shown in Figure 28.1. This solution is conjugate and analytical.

In classical statistics, conjugate priors are highly valued because they yield closed-form posterior distributions. For example, a Beta prior combined with a Binomial likelihood yields a Beta posterior, and a Gamma prior with

a Poisson likelihood yields a Gamma posterior. In BDL, however, conjugate updates are generally unavailable due to the highly non-linear nature of neural networks, where weights are passed through multiple layers of non-linear activations. An exception is last-layer BNN inference, where one can perform exact conjugate Gaussian-Gaussian updates on the final weights by treating the hidden activations of the penultimate layer as fixed features.

An *informative prior* guides the model with subjective domain knowledge, balancing the data’s information and the prior. However, in modern contexts like machine learning and high-dimensional statistics, priors often serve a broader range of purposes beyond encoding prior knowledge¹. When such knowledge is unavailable or neutrality is preferred, *uninformative priors*, such as near-flat or uniform distributions, minimise subjective influence. Similarly, *reference priors* are designed to maximise the difference between the prior and the resulting posterior, letting the data primarily drive inference. In advanced applications of BDL, priors are often selected for practical purposes, such as *convenience priors* for computational simplicity, *regularisation priors* to prevent overfitting, or *sparsity-inducing priors* to promote simpler model structures.

This chapter builds on these ideas, exploring the statistical properties of priors used in BDL. We examine their mathematical formulations, practical roles, and how they shape model behaviour, equipping the reader with tools to design effective BDL models. Each subsection discusses a specific prior type, with examples and visualisations to illustrate their impact. Accompanying figures show the prior distributions and the corresponding prior predictive distributions, which reveal the range of functions a deep learning model might produce before observing data.

28.1 Introductory overview of priors for BDL

In this section, we introduce the fundamental priors employed in Bayesian deep learning, exploring their statistical properties and their impact on model behaviour. For each prior, we provide illustrations of the *prior predictive distribution*, which reflects the range of possible model outputs before observing data. These visualisations are based on a neural network with two hidden layers, each containing two neurons, and logistic (sigmoid) activation functions. While the properties of prior predictive distributions may vary with changes in network depth, width, or activation functions, we focus on this simple architecture to build intuition.

28.1.1 Informative Priors

Informative priors encode specific domain knowledge about the parameters θ , as illustrated in the student height example (Figure 28.1), where priors for average heights of first- and third-grade students were based on past studies. In BDL, “manually specified” informative priors are rare due to the complexity of the models, yet they can still be sometimes applied to specific parameters, such as the bias term θ_0 in the output layer, to reflect expected model outputs at a baseline (e.g. all other input neurons to the last layer should be close to 0 for a chosen baseline combination of covariates). For example, in a BDL model predicting house prices in a city, we might use a one-dimensional Gaussian prior for $\theta_0^{(L)}$:

$$p(\theta_0^{(L)}) = \mathcal{N}(\theta_0^{(L)}; \mu_0, \sigma_0^2),$$

where μ_0 is the average house price (e.g., 500,000 USD) based on market data, and a small σ_0^2 (e.g., 160,000) reflects the level of confidence in this estimate. A strong informative prior (small σ_0^2) dominates the data’s influence (likelihood), while a weak one (larger σ_0^2) balances data and prior information.

Another way to incorporate domain knowledge in BDL is through *spike-and-slab priors* [1190] for weights associated with specific covariates in the input layer. These priors are informative by design when we know certain inputs are more relevant than others. For a weight $\theta_{ij}^{(1)}$ in the first layer corresponding to covariate x_i , the prior takes the form:

$$p(\theta_{ij}^{(1)} | \gamma_i) = \gamma_i \mathcal{N}(\theta_{ij}^{(1)}; \mu_i, \sigma_i^2) + (1 - \gamma_i) \delta_0(\theta_{ij}^{(1)}), \quad \gamma_i \sim \text{Bernoulli}(\rho_i),$$

where $\delta_0(\cdot)$ is a point mass at zero (the “spike”) that switches off covariate x_i when $\gamma_i = 0$, while the Gaussian component $\mathcal{N}(\theta_{ij}^{(1)}; \mu_i, \sigma_i^2)$ (the “slab”) keeps it active when $\gamma_i = 1$. The inclusion probability ρ_i reflects our prior belief in the importance of covariate x_i . For instance, in a model predicting heart disease risk from age, blood pressure, and cholesterol, one might assign $\rho_i = 0.9$ to cholesterol and a lower $\rho_i = 0.1$ to a less relevant variable such as eye colour, as illustrated in Figure 28.2.

Specifying informative priors for all parameters in BDL is a non-trivial task, as the high-dimensional and complex nature of neural networks makes manual prior elicitation practically impossible. While informative

¹There still is a lot of philosophical debate on whether this is acceptable from a puristic Bayesian perspective. We will not dive into this debate and rather introduce different concepts to the reader in the context of BDL.

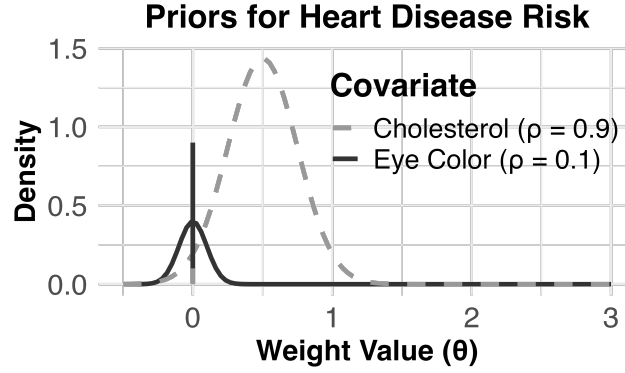


Figure 28.2: Informative spike-and-slab prior density (left) and the corresponding prior predictive distributions (right) for input layer weights under different inclusion probabilities.

priors, such as Gaussian priors for bias terms, can encode domain knowledge for specific parameters, scaling this approach to entire networks is hardly ever feasible. To address this challenge, techniques like pretraining prior hyperparameters, which leverage knowledge from pre-existing models or synthetic data (discussed further in Chapter 30), and function-space priors, which impose structure on the model’s output functions (explored in detail in Chapter 31), offer promising solutions. These approaches mitigate the difficulties of manual prior specification, at least to some extent, and provide practical strategies for effective prior design in BDL.

28.1.2 Uninformative, Reference, and Objective Priors

When prior knowledge is absent or neutrality is desired, statisticians employ objective or *uninformative priors* to minimise subjective influence, letting the data dominate inference. In practice, a common choice is a uniform prior for each parameter:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P \text{Uniform}(\theta_i; a, b),$$

where $[a, b]$ is a wide interval (e.g., $[-10, 10]$). This results in a joint prior that is uniform over a hyper-rectangle $[a, b]^P$, with each marginal distribution $p(\theta_i)$ being uniform, suggesting neutrality for individual parameters.

However, marginally uniform priors do not always imply an uninformative joint prior [564], especially in high-dimensional BDL models. The independence assumption can impose unintended structure, as parameters like weights in deep learning models often interact. Moreover, the joint prior can become concentrated in certain regions, undermining its uninformative nature. For instance, if $[a, b] = [-1, 1]$ is chosen to be narrow, the joint prior restricts all weights to a small hyper-rectangle, concentrating probability mass and limiting the model’s ability to capture complex functions, such as non-linear patterns in a neural network. Similarly, in high-dimensional settings with large P , the product of uniform priors can lead to a prior predictive distribution (the range of possible model outputs) that concentrates around specific function classes, such as near-zero outputs if the network’s activation functions (e.g., ReLU) amplify small weights. This concentration can bias inference, making the prior less uninformative than intended and resulting in quite concentrated prior predictive distributions. Additionally, there is a long-standing statistical debate regarding whether a bounded uniform prior is truly uninformative; strictly speaking, a uniform prior is only uninformative in its improper version (over the entire real line $\mathbb{R}$), which is non-normalisable, potentially causing unstable inference or improper posteriors in BDL’s high-dimensional context. Furthermore, flat or uniform priors are not invariant under parameter transformations, meaning a prior that appears uniform on one scale may encode strong beliefs on another.

To address this lack of invariance, statisticians construct *objective priors*. The classic example is Jeffreys prior [840], which is defined as proportional to the square root of the determinant of the Fisher information matrix. Jeffreys prior is invariant under reparameterisation, meaning the same prior beliefs are represented regardless of how the model is parameterised. In multi-parameter settings, this idea is generalised by *reference priors* [130], which are designed to maximise the divergence (e.g., KL divergence) from the prior to the posterior as the sample size grows, ensuring the data’s influence is maximised. Reference priors account for the model’s structure and parameter interactions, making them more robust for complex models. For example, in a BDL model, a reference prior might adjust for the network’s architecture to avoid the biases introduced by uniform priors. However, the computational complexity of computing Jeffreys or reference priors (which requires evaluating the Fisher information of a neural

network) limits their use in BDL, where models often have millions of parameters. In practice, uninformative priors like uniform distributions are rarely used since despite simplicity they require careful selection of $[a, b]$ to avoid unintended concentration or instability.

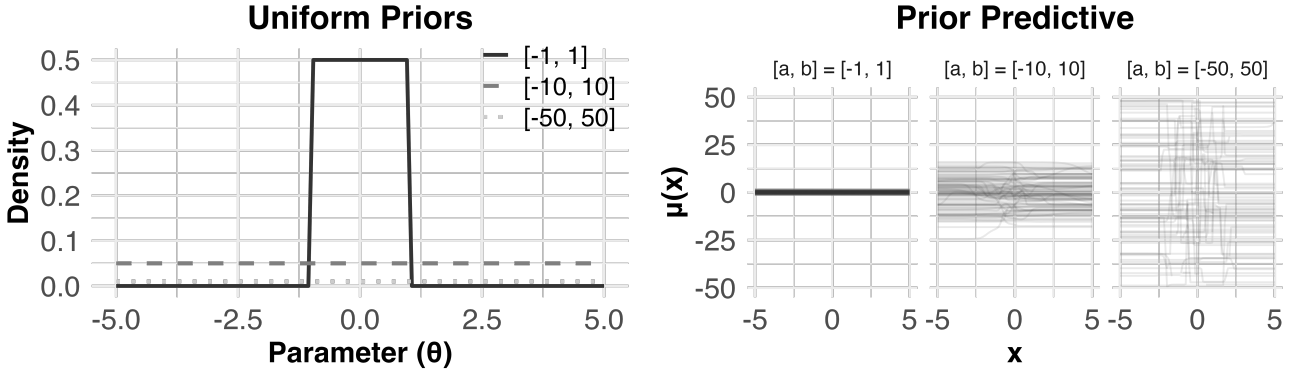


Figure 28.3: Uniform priors for single parameters (left) and the corresponding prior predictive distributions (right) under varying intervals, showing how narrow bounds can restrict the function space.

The effect of different intervals is illustrated in Figure 28.3, which shows the density of uniform priors for three intervals: $[-1, 1]$, $[-10, 10]$, and $[-50, 50]$ and their impact on the prior predictive distribution. Sample trajectories for a BDL regression model under these priors are not uniform and can be strongly concentrated in specific regions, cutting the support for the prior predictive distribution and highlighting our discussion above. This prevents the prior predictive from being non-informative on one hand, and on the other, limits its support, which makes uniform priors a poor tool both for variational methods and for regularisation in BDL. A related but distinct approach to an uninformative prior is the limiting Gaussian $\mathcal{N}(\theta_i; 0, \sigma^2)$ as $\sigma^2 \rightarrow \infty$, which behaves similarly to a flat prior over any bounded region of interest and avoids the boundary effects of uniform distributions. In the next subsection, we shall discuss other prior alternatives for BDL.

28.1.3 ℓ_2 Regularisation Priors

Regularisation priors prevent overfitting by constraining model complexity. A standard choice in BDL is the zero-centred isotropic Gaussian prior for all weights in the model:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P \mathcal{N}(\theta_i; 0, \sigma^2).$$

The popularity of this prior is not just in its simplicity, it actually corresponds to ℓ_2 regularisation in maximum a posteriori (MAP) estimation, as derived below. The variance σ^2 controls regularisation strength: smaller σ^2 shrinks parameters toward zero, simplifying the model, while larger σ^2 allows more flexibility. To derive the link to ℓ_2 -regularisation, consider a BDL model with parameters $\boldsymbol{\theta}$. In MAP estimation, we maximise the log-posterior:

$$\log p(\boldsymbol{\theta} \mid \mathcal{D}) = \log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{constant}.$$

For the i.i.d. Gaussian prior, the log-prior is:

$$\log p(\boldsymbol{\theta}) = \sum_{i=1}^P \log \mathcal{N}(\theta_i; 0, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{i=1}^P \theta_i^2 - \frac{P}{2} \log(2\pi\sigma^2).$$

Thus, the log-posterior is:

$$\log p(\boldsymbol{\theta} \mid \mathcal{D}) = \log p(\mathcal{D} \mid \boldsymbol{\theta}) - \frac{1}{2\sigma^2} \sum_{i=1}^P \theta_i^2 + \text{constant}.$$

The term $\frac{1}{2\sigma^2} \sum_{i=1}^P \theta_i^2$ is the ℓ_2 -regularisation penalty, with $\lambda = \frac{1}{2\sigma^2}$. A smaller σ^2 increases λ , enforcing stronger regularisation.

To illustrate the role of the prior variance σ^2 in regularisation within BDL, consider three cases: $\sigma^2 = 0.02$, $\sigma^2 = 1$, and $\sigma^2 = 50$, corresponding to strong, moderate, and weak regularisation, respectively. The regularisation

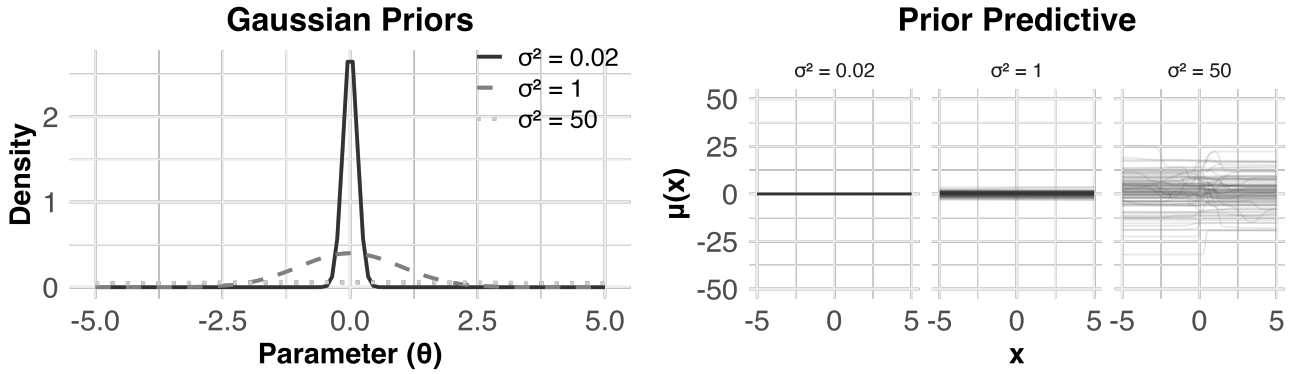


Figure 28.4: Gaussian priors for single parameters (left) and the corresponding prior predictive distributions (right) under the baseline architecture and varying σ^2 .

parameter $\lambda = \frac{1}{2\sigma^2}$ governs the penalty on model weights. The small σ^2 of 0.02 yields a large $\lambda = \frac{1}{2 \cdot 0.02} = 25$, heavily penalising large weights, which constrains the model to produce simple, near-linear functions, particularly for activation functions like the logistic one, which is approximately linear near zero. Conversely, a large σ^2 of 50 results in a small $\lambda = \frac{1}{2 \cdot 50} = 0.01$, allowing larger weights and enabling complex, non-linear functions, albeit with an increased risk of overfitting. A moderate σ^2 of 1 lies between these extremes, with $\lambda = 0.5$, permitting some flexibility while controlling model complexity.

These effects manifest in the *prior predictive distribution*, which captures the range of possible model outputs before observing data. In a BDL regression model, a small σ^2 concentrates prior predictive outputs near zero, yielding smooth, constrained functions that are close to linear or even near-constant, while a large σ^2 permits highly variable, non-linear outputs, accommodating diverse patterns. Notably, the support of the prior predictive distribution remains unconstrained across all cases, making Gaussian priors versatile for robust regularisation. Figure 28.4 illustrates this effect, highlighting the critical role of σ^2 in shaping model behaviour.

The above discussion assumes a neural network with two hidden layers, each with two neurons, and a logistic activation function. However, prior predictive behaviour depends on architectural choices, particularly activation functions. While regularisation persists if the activation is approximately linear near zero, the specific functional form of outputs varies with activation properties, such as boundedness. Figure 28.5 compares prior predictive distributions for tanh and ReLU activations in the same architecture, showcasing distinct behaviours.

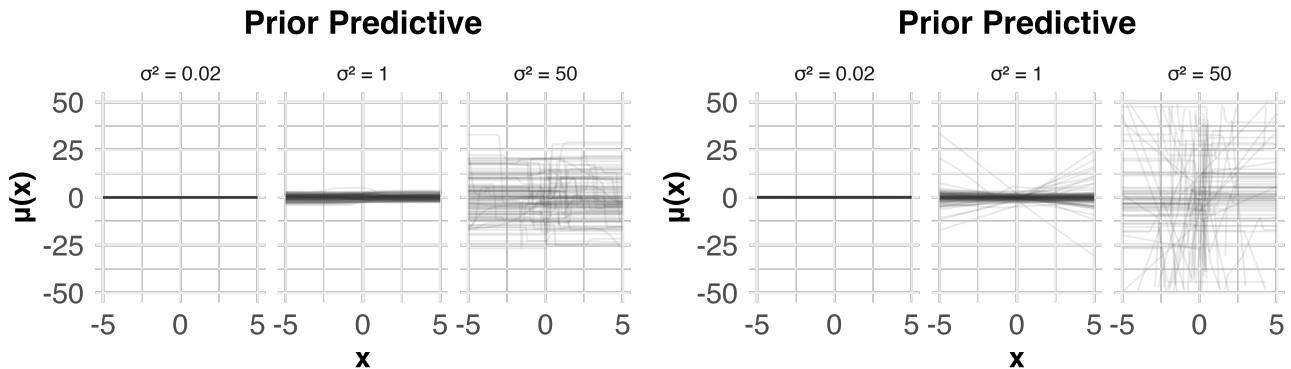


Figure 28.5: Prior predictive distributions for tanh (left) and ReLU (right) activations.

Such architectural dependencies introduce further risks of prior misspecification. To mitigate this, it is recommended to conduct prior predictive checks on simplified versions of the model.

As an example of a well-specified model and a misspecified one, consider a classification problem using our baseline BDL model with two input covariates and Bernoulli-distributed responses. Varying the prior variance σ^2 alters the decision boundary based on the posterior predictive ², transitioning from a near-constant model

²Obtained with a 1000 HMC iterations on two chains, computed by Stan, yet inference details are not of central interest in this chapter, hence omitted.

(small $\sigma^2 = 0.02$), to close to linear (moderate $\sigma^2 = 1$), to non-linear boundaries (large $\sigma^2 = 50$). Figure 28.6 illustrates this property, underscoring that a naive Bayesian approach offers no inherent guarantees of optimality or robustness in the posterior predictive distribution.

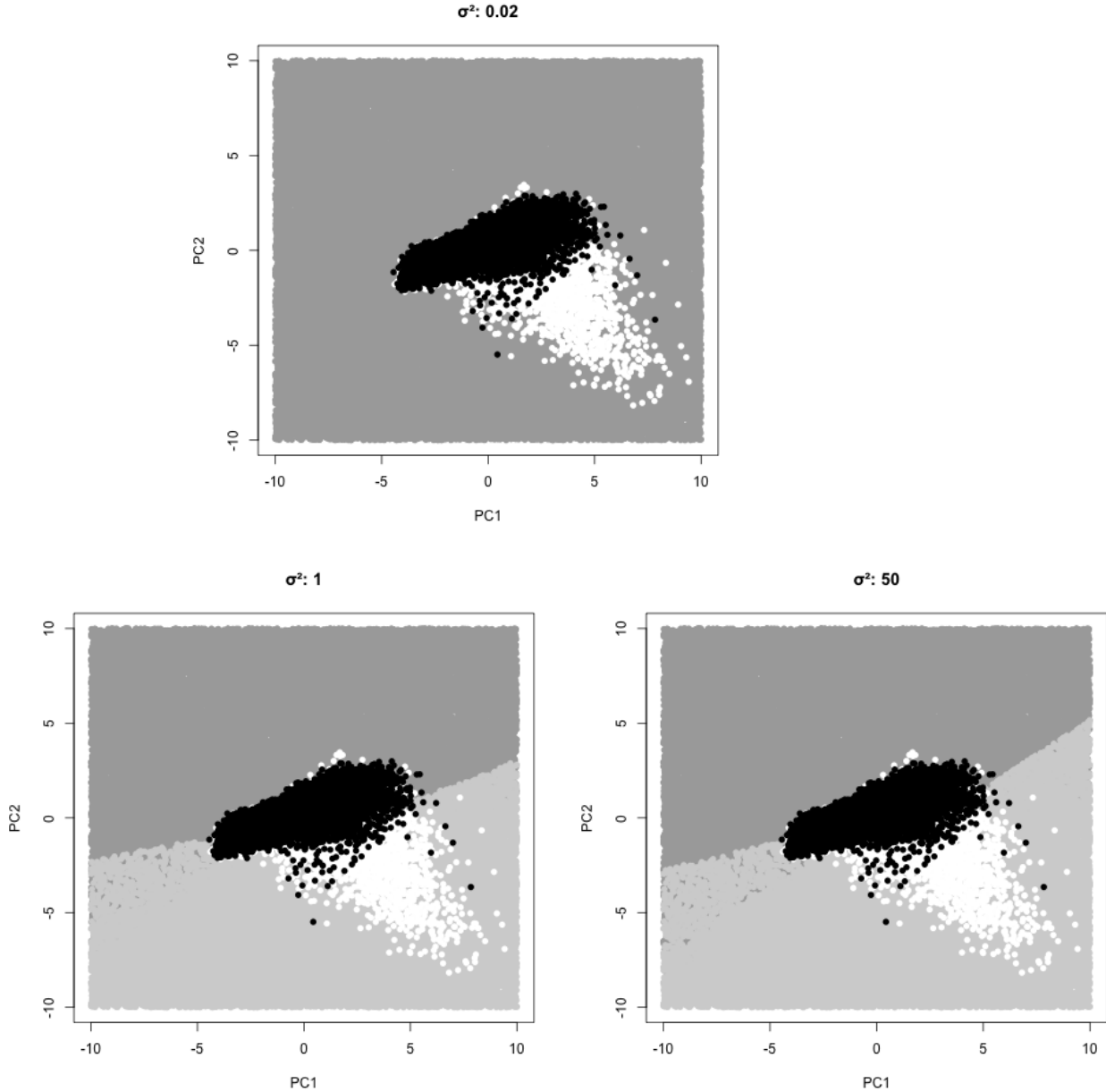


Figure 28.6: Decision boundaries based on the posterior predictive of a BDL classification model with prior variances $\sigma^2 = 0.02$, 1, and 50.

As we have seen already, poorly chosen σ^2 in Gaussian priors can lead to severe model misspecification, resulting in distorted, misspecified posteriors, a phenomenon known as the *cold posterior effect* [549, 1742]. Blindly adopting a standard Gaussian prior is ill-advised. To address this, σ^2 can be tuned via empirical Bayes or assigned a hyperprior, such as $\sigma^2 \sim \text{Inv-Gamma}(\alpha, \beta)$ [968] resulting in a robust t-distributed marginal prior. In contrast, frequentist ℓ_2 -regularisation typically selects λ through cross-validation or model selection criteria.³

Another extension of i.i.d. Gaussian priors is the multivariate Gaussian prior, which accounts for parameter dependencies via a covariance structure:

$$p(\boldsymbol{\theta}) = \mathcal{N}_P(\boldsymbol{\theta}; \mathbf{0}, \boldsymbol{\Sigma}),$$

³Hyperparameter optimisation can also tune σ^2 directly, but this approach deviates from a strictly Bayesian perspective.

with the possibility to introduce an additional inverse-Wishart hyperprior on Σ . Such priors (at least for sparse covariance structures) may be practical for large-scale BDL as they both provide regularisation and allow for handling dependencies between the parameters.

28.1.4 Sparsity-Inducing and Variable Selection Priors

Sparsity-inducing priors encourage model parameters to be zero or near-zero, which helps simplify the model structure, prevent overfitting, and improve generalisation. In BDL, two broad classes of sparsity are commonly distinguished. *Soft-sparsity*, achieved through continuous shrinkage priors such as the Laplace, Horseshoe, and scale mixtures of Gaussians, concentrates probability mass around zero but produces almost surely non-zero (non-sparse) posterior samples. These priors act as shrinkage mechanisms that pull uninformative weights toward zero while allowing large weights for genuinely important features, thanks to their heavy tails. The Laplace prior, for instance, yields exactly sparse MAP estimates (analogous to the frequentist Lasso [1607]), but posterior samples remain non-sparse. *Hard-sparsity*, in contrast, is achieved through priors that place a discrete probability mass exactly at zero, such as spike-and-slab priors [607, 1190]. These priors allow parameters to be exactly zero in the posterior, performing explicit variable selection and zeroing out weights, which is particularly valuable in BDL for weight pruning, network compression, and feature selection.

The Laplace prior is a common continuous shrinkage prior:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P \text{Laplace}(\theta_i; 0, b),$$

which corresponds to ℓ_1 -regularisation in MAP estimation with $\lambda = b^{-1}$ [1295, 1761]. The scale parameter b controls the degree of shrinkage, similar to σ^2 in the Gaussian prior.

Other continuous options include the scale mixture of Gaussians [174]:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P (\pi \mathcal{N}(\theta_i; 0, \sigma_1^2) + (1 - \pi) \mathcal{N}(\theta_i; 0, \sigma_2^2)),$$

with $\sigma_1^2 > \sigma_2^2$ and $\sigma_2^2 \ll 1$, which places a narrow Gaussian near zero to shrink small weights and a wide Gaussian to accommodate larger weights. Alternatively, one can use the horseshoe prior [243]:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P \mathcal{N}(\theta_i; 0, \sigma^2 \tau_i^2),$$

where $\tau_i \sim \text{C}^+(0, 1)$ is a half-Cauchy local shrinkage parameter. Both Laplace and Horseshoe priors are heavy-tailed, allowing them to shrink noise parameters aggressively while leaving large, informative parameters virtually unaffected. This heavy-tailed property makes them robust to misspecification and helps mitigate the cold posterior effect in BDL [549].

For hard-sparsity and explicit variable selection, the spike-and-slab prior [607, 1190] can be applied to every weight $\theta_{ij}^{(l)}$ in the network:

$$p(\theta_{ij}^{(l)} | \gamma_{ij}^{(l)}) = \gamma_{ij}^{(l)} \mathcal{N}(\theta_{ij}^{(l)}; 0, \sigma_1^2) + (1 - \gamma_{ij}^{(l)}) \delta_0(\theta_{ij}^{(l)}), \quad \gamma_{ij}^{(l)} \sim \text{Bernoulli}(\rho),$$

where $\delta_0(\cdot)$ is a Dirac delta point mass at zero (the “spike”) that sets weight $\theta_{ij}^{(l)}$ to exactly zero when $\gamma_{ij}^{(l)} = 0$, while the Gaussian component $\mathcal{N}(\theta_{ij}^{(l)}; 0, \sigma_1^2)$ (the “slab”) allows non-zero weights when $\gamma_{ij}^{(l)} = 1$. The inclusion probability ρ governs the overall network sparsity: a low value encourages aggressive pruning, while a high value retains most weights. When the inclusion probabilities vary by weight or layer, they can encode structural domain knowledge, as illustrated in the informative prior example in Figure 28.2. The spike-and-slab prior is closely related to the model priors discussed in Section 28.1.5.

Log-uniform priors, either improper [1196]:

$$p(\boldsymbol{\theta}) = \prod_{i=1}^P \text{LogU}_\infty(\theta_i) \propto \prod_{i=1}^P \frac{1}{\theta_i}$$

or proper [1231]:

$$p(\boldsymbol{\theta}) \propto \prod_{i=1}^P \text{LogU}_\infty(\theta_i) \text{I}_{[a,b]}(\log \theta_i),$$

also promote sparsity by placing heavy mass near zero, though they remain less studied in classical statistics.

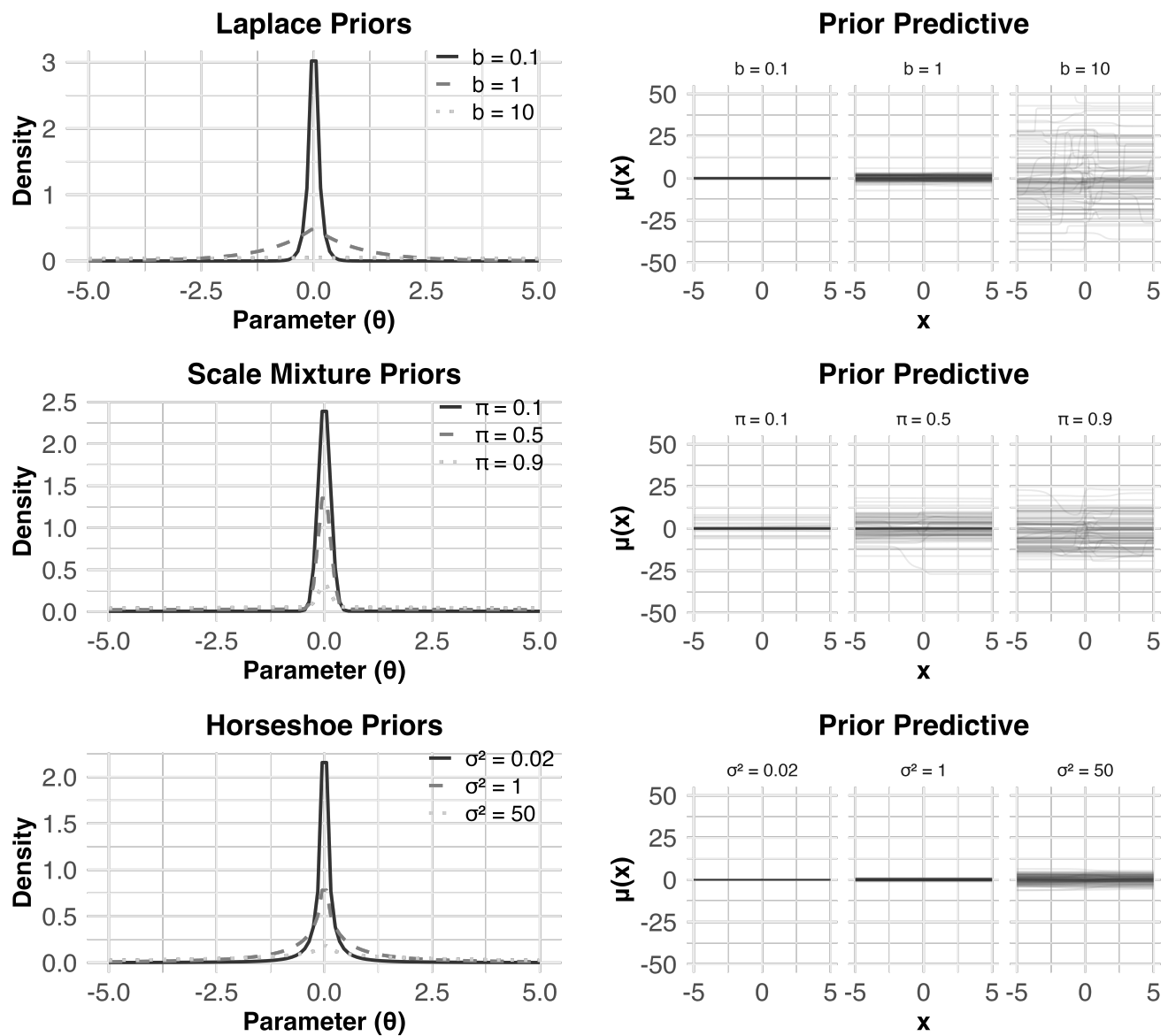


Figure 28.7: Continuous sparsity-inducing prior densities for single parameters (left column) and the corresponding prior predictive distributions (right column) under the baseline architecture: Laplace (top row), scale mixture of Gaussians (middle row), and horseshoe (bottom row) priors under varying hyperparameters.

28.1.5 Model Priors and Bayesian Model Selection and Averaging

In all previous examples, we restricted ourselves to a single BDL model. In this subsection, we allow for multiple BDL models and thus capture model uncertainty. In Bayesian model averaging (BMA), multiple models $\{\mathcal{M}_1, \dots, \mathcal{M}_t\}$ are considered, requiring priors $p(\mathcal{M})$. We assume prior mass for all these models $p(\mathcal{M}_i), i \in \{1, \dots, t\}$ and define parameter priors conditionally on the respective models $p(\boldsymbol{\theta}_i | \mathcal{M}_i), i \in \{1, \dots, t\}$. Generally speaking, models could be competing neural networks of different types (such as dense, convolutional, or residual ones) with varying numbers of layers and hidden units, but for simplicity we will only consider models defined by different weight inclusion patterns [801, 1521]. The prior over the model indicator vector $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_P)$ takes the same structural form as the spike-and-slab inclusion prior introduced in Section 28.1.4, but is now interpreted as selecting entire model structures rather than individual weights. In this context, Bernoulli inclusion priors are commonly used:

$$p(\mathcal{M}) = \prod_{i=1}^P \text{Bernoulli}(\gamma_i; \rho_i),$$

where $\gamma_i \in \{0, 1\}$ indicates weight inclusion, and ρ_i is the inclusion probability. Beta-Binomial priors [802] are an alternative:

$$p(\mathcal{M}) \propto \prod_{i=1}^P \text{BetaBinomial}(\gamma_i; 1, a_i, b_i),$$

which correspond to a Beta-Bernoulli hierarchy where the success probabilities ρ_i are integrated out under a Beta prior, helping control the overall model size. These priors can encode domain knowledge, such as assigning higher inclusion probabilities for input weights in a BDL model for medical diagnosis. Parameter priors are then specified conditionally on the model: $p(\boldsymbol{\theta} | \mathcal{M})$, as discussed in the previous subsections.

Inference on the posterior $p(\mathcal{M} | \mathcal{D})$ allows us to perform model selection through the median probability model [95], which often leads to extreme sparsification [787], or to perform global explanations based on posterior inclusion probabilities [787].

28.2 Summary and Outlook

This chapter introduced the fundamental role of priors in Bayesian deep learning, progressing from classical informative and conjugate formulations to the regularisation, shrinkage, and variable selection priors commonly applied in modern deep networks. While parameter-space prior elicitation remains challenging, understanding how weight-space choices shape the prior predictive distribution is vital for building robust BDL models. More advanced priors designed to scale to deep structures or handle dependencies have been proposed, including priors that model weight correlations (e.g. Rudner et al. [1434]) and subnetwork priors that isolate key parameters (e.g. Sell and Singh [1488]).

The subsequent chapters in this part develop these concepts further. In Chapter 29, we explore the statistical and mathematical properties of parameter priors as they propagate through hidden units and define infinite-width limits. In Chapter 30, we study data-driven informative priors constructed to achieve desired functional properties. Finally, Chapter 31 introduces function-space priors that bypass weight-space pathology by directly defining distributions over functional outputs.

Chapter 29

Statistical properties of priors

The previous introductory Chapter 28 introduced how priors in Bayesian Deep Learning (BDL) shape model behaviour by influencing the prior and posterior distributions over parameters and functions. While informative priors (see Chapter 30) can encode domain knowledge, their use in BDL is limited due to model complexity. Instead, priors often serve practical roles such as regularisation, promoting sparsity, or enabling efficient inference. Chapter 28 emphasised the importance of understanding prior predictive distributions and illustrated how architectural choices interact with different prior types. This chapter further studies how priors over neural network parameters induce priors over hidden layer units. The main focus is on deriving distributional properties and their potential effects on neural network training. Further, Chapter 31 explores priors in function space, another way to study the influence of priors on the models. Function-space priors are often interpretable but analytically intractable. Chapter 31 highlights pathologies that arise from common parameter priors and proposes matching neural network-induced function priors to target function-space distributions such as Gaussian processes using sample-based objectives like Wasserstein distance or Maximum Mean Discrepancy.

29.1 Introduction

Priors, probability distributions that encapsulate our initial beliefs about model parameters, lie at the heart of Bayesian neural networks (BNNs). The choice of priors is not merely a theoretical consideration but has profound implications for the statistical properties of the resulting models. The theoretical study of priors in BNNs dates back to the foundational work of Neal [1226], who showed that single-hidden layer neural networks with *i.i.d.* Gaussian weights converge to a Gaussian process as the number of neurons of the hidden layer tends to infinity (the so-called infinite-width limit), see Theorem 29.2. After providing an overview of the statistical properties of priors in Bayesian inference and, more specifically, in BNNs, this chapter is organised as follows. Sections 29.2 and 29.3 examine single-hidden-layer and deep neural networks, respectively. Sections 29.4 and 29.5 consider the role of different priors and architectures. Finally, the impact of the prior on the posterior is discussed in Section 29.6.

29.1.1 Simple statistical properties of priors

The core principle of the Bayesian framework is to represent uncertainty about model parameters and predictions using probability distributions. Unlike non-Bayesian or frequentist methods, which treat parameters as fixed but unknown quantities, the Bayesian perspective considers parameters as random variables. Bayesian inference refers to the inductive process of updating beliefs in light of observed data via Bayes' rule. This choice is not arbitrary: it is grounded in rigorous mathematical foundations, notably from the works of Cox and Savage [339, 1463].

We adopt the following notation: let $\mathcal{D} = (X, Y)$ denote a training dataset of size N , assumed to be generated by an underlying *sampling model* or *likelihood* $p(\mathcal{D}|\theta)$. The parameters θ lie in a parameter space Θ , typically a subset of the Euclidean space $\mathbb{R}^P$. The *prior distribution* $p(\theta)$ encodes our beliefs about the parameters θ before observing any data, and constitutes the central theme of this chapter.

Specifying both the prior $p(\theta)$ and the likelihood $p(\mathcal{D}|\theta)$ defines the *joint distribution* of the parameters and data via the product rule: $p(\theta, \mathcal{D}) = p(\theta)p(\mathcal{D}|\theta)$. Bayes' rule combines the prior and the likelihood to produce the *posterior distribution* $p(\theta|\mathcal{D})$:

$$p(\theta|\mathcal{D}) = \frac{p(\theta)p(\mathcal{D}|\theta)}{p(\mathcal{D})}. \quad (29.1)$$

Here, the denominator $p(\mathcal{D})$, known as the model *evidence* or *marginal likelihood*, serves as the normalisation constant.

Bayes' rule (29.1) provides a principled mechanism for updating our beliefs in light of new data, but it offers no guidance on how those beliefs, encoded in the prior, should be initially specified. In practice, prior choices are often influenced by computational convenience. A notable example is the use of conjugate priors: a prior is said to be conjugate to a likelihood if the resulting posterior belongs to the same parametric family. Classic instances of conjugate prior-model pairs include the Gaussian-Gaussian, Beta-Binomial, and Gamma-Poisson.

The issue of prior choice often leads to the question: how informative is the prior? This introduces a distinction between objective and subjective priors. For example, Jeffreys' prior, defined as proportional to the square root of the determinant of the Fisher information matrix, is considered objective due to its invariance under reparameterisation. See Lee [999] and Nalisnick and Smyth [1222] for studies on Jeffreys' priors in BNNs. Objective or uninformative priors are frequently improper, meaning they do not integrate to a finite total probability (e.g., a uniform prior over an unbounded space). Surprisingly, such priors are widely used in Bayesian analysis, provided the resulting posterior is proper and thus suitable for inference. However, improper priors cannot be used to compute prior predictive distributions or Bayes factors, as these too would be improper.

In contrast, subjective priors lie at the core of the Bayesian philosophy, encoding expert beliefs or prior knowledge. The process of eliciting such priors is nuanced and challenging; see, for example, Mikkola et al. [1173] for a survey.

The challenge of specifying a prior becomes even more pronounced as models grow in complexity, when parameters lack clear interpretation, or when the parameter space is high-dimensional. In such settings, designing a prior that appropriately covers the space becomes non-trivial, and exact computation of the posterior may be infeasible. In these cases, the quality of Bayesian inference relies heavily on both the approximations made for computational tractability and the form of the prior distribution itself.

29.1.2 Statistical properties of priors in BNNs

The Gaussian distribution is a common and default choice of prior in Bayesian neural networks. Looking for a Maximum a Posteriori (MAP) estimate of such a Bayesian model is equivalent to training a standard neural network under weighted L_2 regularisation. There is currently neither theoretical nor empirical evidence that Gaussian priors are preferable to alternative choices [1213]. Yet, its well-studied mathematical properties lead to Gaussian distributions being the default prior choice in BNNs.

Arguably, the prior that matters the most from a practitioner's point of view is the prior induced in function-space, not in parameter space or weight-space [1767], as the induced prior seen at the function level can provide insight into the typical behaviour of the output functions. For reviews on priors in Bayesian deep learning models, refer to Arbel et al. [43], Fortuin [547], Nalisnick [1223], as well as Chapter 28.

29.1.3 Width and depth limits of BNNs and their implications

While inference in Bayesian neural networks is computationally limited to finite networks, the statistical properties of Bayesian neural networks vary depending on whether one considers finite networks or their infinite width or infinite depth limits.

In the finite width regime, BNNs maintain a finite number of parameters. Integrating Bayesian inference principles allows them to represent uncertainty about the output function of interest by considering a distribution over the network parameters. This approach is crucial in scenarios where overconfidence can lead to detrimental outcomes, such as in identifying rare diseases from medical images or predicting financial market trends.

In the infinite width limit, many BNNs exhibit a theoretical convergence to Gaussian processes (GPs), in the sense that, as the width of the network approaches infinity, the prior distribution over functions induced by a BNN approaches that of a GP. The infinite width perspective illuminates the behaviour of neural networks at scale, offering insights into why wide networks perform well in practice, and aids in the understanding and development of network architectures and priors that can leverage this convergence for improved generalisation and robustness.

In the context of infinite depth limits, BNNs are able to capture increasingly complex functions through deeper architectures. As networks grow deeper, they have the potential to model intricate hierarchical structures within the data. In the infinite depth regime, these networks can exhibit recursive feature transformations that resemble deep GPs [361], see Chapter 25. This allows for more expressive power and the capacity to model data with rich, layered patterns. Theoretical analysis in this regime offers insights into phenomena like the interplay between depth and nonlinearity, leading to smoother functions and better generalisation. However, maintaining computational tractability and avoiding issues such as vanishing gradients become essential challenges. Exploring infinite depth

networks provides a deeper understanding of how neural networks can effectively capture and utilise complex dependencies in data.

Furthermore, the relationship between finite BNNs and their infinite limits also simplifies theoretical analyses and provides a framework for studying the impact of architectural choices, such as depth and activation functions, through a probabilistic lens. This dual perspective makes BNN priors a rich field of study, which we are investigating in the following pages.

29.1.4 Outline

The rest of this chapter is organised as follows. Section 29.2 studies how priors propagate through shallow neural networks (with a single hidden layer), showing the transition from sub-Gaussian first-layer pre-activations to sub-exponential second-layer pre-activations, the “uncorrelated yet dependent” phenomenon among units, and the Gaussian-process limit for shallow nets. Section 29.3 develops the deep case, proving sub-Weibull tails with index increasing in depth, giving exact Meijer-G characterisations for linear/ReLU BNNs, and extending GP limits (including variants yielding Student-t processes under hyperpriors). Figure 29.1 provides a schematic illustration of the material in Section 29.2 and Section 29.3. Section 29.4 contrasts prior families and then details two examples: α -stable priors that preserve heavy tails in infinite width via stable-process limits, and trace-class priors that maintain identifiable hidden units in the infinite-width limit. Section 29.5 examines how activation assumptions (extended envelope) and architectural motifs (residual connections, convolutions, and neural ODEs) reshape unit and function-space priors, typically modulating tail behaviour and correlations. Finally, Section 29.6 analyses how these priors affect posteriors, covering contraction rates (including spike-and-slab and scalable alternatives), statistical adaptation, limits of Bernstein–von Mises phenomena in overparameterised settings, and links to deep GP asymptotics and high-dimensional regimes.

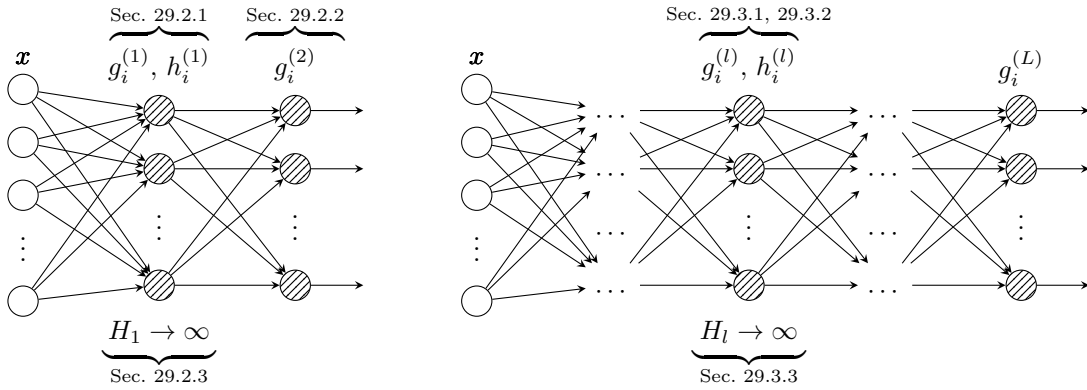


Figure 29.1: Schematic illustration of the material in Section 29.2 and Section 29.3.

29.2 Single-hidden-layer networks

Understanding the distribution of units (pre-activations and post-activations in each layer) under the prior can illuminate how information propagates in an untrained network. This is the focus of a series of works by Vladimirova et al. [1682, 1684, 1685]. The key finding is that unit distributions get progressively heavier-tailed in deeper layers when weights have light-tailed prior distributions, such as Gaussian priors.

29.2.1 First-hidden-layer outputs

Let $x \in \mathbb{R}^D$ be an input vector and assume neural network weights θ follow *i.i.d.* Gaussian distributions $\mathcal{N}(0, \sigma^2)$, and assume, for now, that we have no bias parameters. Then, the pre-activation of the i -th hidden unit in the first layer is a sum of weighted inputs:

$$g_i^{(1)} = \sum_{j=1}^D \theta_{i,j}^{(1)} x_j \sim \mathcal{N}(0, \sigma^2 \|x\|^2), \quad (29.2)$$

due to the linearity of expectation and the independence of weights. This result illustrates a key insight: under Gaussian priors, the input to the nonlinearity in the first layer is itself Gaussian.

However, once a nonlinearity ϕ (e.g., ReLU) is applied to obtain the activation $h_i^{(1)} = \phi(g_i^{(1)})$, the output distribution becomes non-Gaussian. For example, with ReLU, $h_i^{(1)}$ is a half-Gaussian (i.e., a Gaussian clipped below zero). The resulting distribution is sub-Gaussian, meaning that its tails decay at least as fast as Gaussian:

$$\mathbb{P}(|h_i^{(1)}| > t) \leq C \exp(-t^2/K), \quad \text{for any } t \geq 0, \quad (29.3)$$

for some constants $C, K > 0$. This sub-Gaussianity implies that, at the first layer, the prior retains relatively light tails, an important property that shapes the induced prior on functions.

[153] provides closed-form expressions for the first two moments of post-activation units in single-hidden-layer networks with Gaussian weights. These moment characterisations form the basis for bounding tail behaviour and variance growth. However, the restriction to a single hidden layer limits their analysis and does not capture the cumulative non-linear transformations seen in deeper architectures.

29.2.2 Combining the outputs of a single-hidden-layer neural network

Since the outputs of the first hidden layer are non-linear transformations of Gaussians, they are no longer Gaussian. When these are multiplied by a new set of independent Gaussian weights in the second layer (see Figure 29.1), the resulting pre-activations become products of sub-Gaussian and Gaussian variables, i.e. $g_i^{(2)} = \sum_{j=1}^{H_1} \theta_{i,j}^{(2)} h_j^{(1)}$. Vladimirova et al. [1682] shows that such products are sub-exponential with tail bounds of the form:

$$\mathbb{P}(|g_i^{(2)}| > t) \leq C \exp(-t/K), \quad \text{for any } t \geq 0, \quad (29.4)$$

for some constants $C, K > 0$. This heavier tail behaviour marks the transition from sub-Gaussian to sub-exponential distributions and is one of the earliest indicators that deep networks encode increasingly heavy-tailed beliefs under standard Gaussian weight priors.

The previous results provide marginal properties of the prior distribution of hidden units, but their joint distribution is more difficult to characterise as some dependence arises from the second hidden layer, although the units across a layer are uncorrelated [1685]. Theorem 29.1 formalises this property, showing that, under the prior distribution, any two units of any layer are uncorrelated if the weight priors are independent and centred at 0:

Theorem 29.1 (Hidden units dependence, Vladimirova et al. [1685]). *If the prior p on the neural network weights θ is centred, in the sense that $\mathbb{E}_p(\theta) = 0$, and independent (isotropic), then any two units of any layer are uncorrelated a priori. Also, their Kendall's tau and Spearman's rho vanish.*

The proof is left to the reader as an exercise.

For finite-width networks, these dependent and non-Gaussian distributions complicate analytic treatment. Marginal properties alone are insufficient for accurate description, as the function-space prior cannot be characterised by simple moments. Approximate representations, such as the use of sub-Weibull, generalised Weibull or Meijer G -function-based priors (see Section 29.3), aim to capture these richer properties in analytically tractable forms.

29.2.3 Gaussian process limit for shallow neural networks

As shown by Neal [1226], when the number of hidden units H_1 tends to infinity, the sum of independent (or weakly dependent) sub-exponential variables with properly scaled variance converges to a Gaussian via the Central Limit Theorem. Figure 29.2 illustrates the width limit for both the single-hidden-layer case (top row) and its generalisation to deeper networks, Theorem 29.4, due to Matthews et al. [386]. The induced prior on the function space then becomes a Gaussian process, with a covariance kernel determined by the activation function and weight variance.

Theorem 29.2 (Neal [1226]). *Consider a Bayesian neural network with one hidden layer with H_1 nodes, and let each weight follow a $\mathcal{N}(0, 1/H_1)$ distribution. If the post-activations have bounded variance, then the BNN output converges to a zero-mean Gaussian process as H_1 tends to ∞ .*

The proof is left to the reader as an exercise.

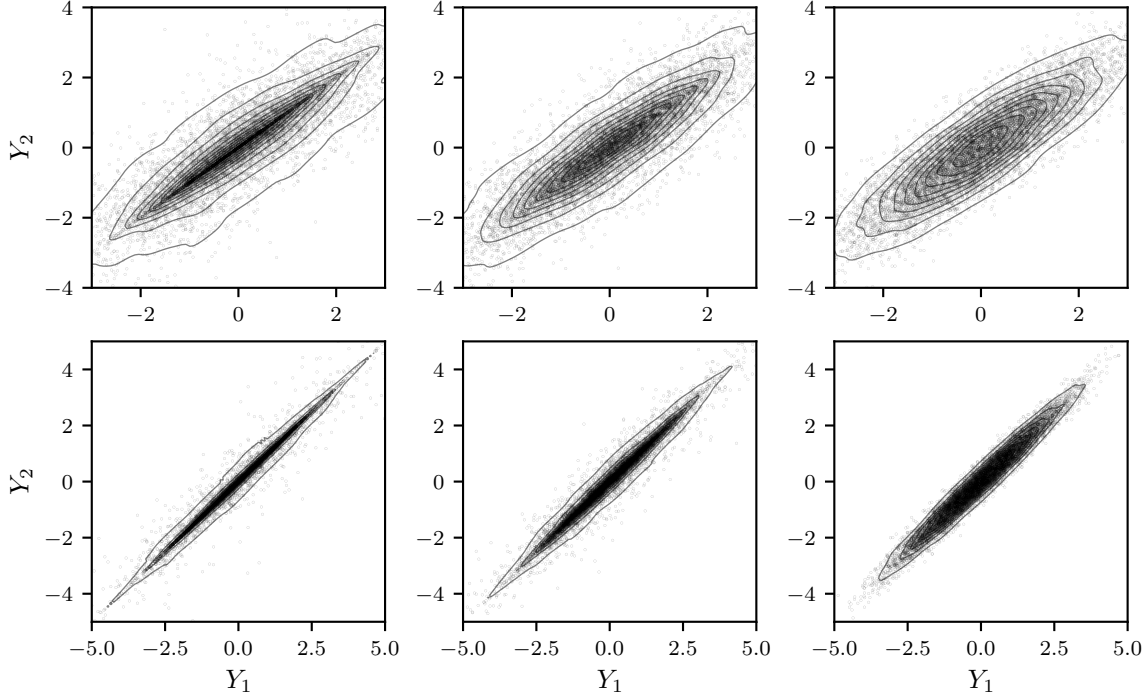


Figure 29.2: Verification of the Gaussian process limit of single-hidden-layer BNNs (Theorem 29.2) and its generalisation to deeper neural networks (Theorem 29.4). Joint distribution at two locations for BNNs of different sizes: one hidden layer (top row) and three hidden layers (bottom row) with 2 (left column), 5 (middle), and 1,000 (right) hidden nodes per layer. Kernel density plots are shown for the respective distributions based on 10,000 samples.

29.3 Deep neural networks

As we saw, some dependence between hidden units arises from the second hidden layer, although the units across a layer are uncorrelated. Thus, as also shown in the previous section for two hidden layers, the marginal distribution of l -th layer hidden units is difficult to characterise. Vladimirova et al. [1682, 1684] described unit priors through sub-Weibull and Weibull-tail distributions on the units using the dependence characterisation between the hidden units. Noci et al. [1246], Zavatone-Veth et al. [1833] provide a precise description of the hidden units prior distribution of both linear and ReLU BNNs. One implication of these works is that the preactivation prior is radial. More specifically, if $\mathbf{g}$ denotes the preactivations of some layer, then the prior $p(\mathbf{g})$ can be written as $p(\|\mathbf{g}\|)$. This property enables [1246, 1832] to obtain exact expressions for hidden units priors in terms of so-called Meijer G -functions by exploiting the relationship between the Fourier transforms of radial functions and the Hankel transform.

29.3.1 Sub-Weibull characterisation

The hidden units distributions can be characterised as sub-Weibull distributions, a generalisation of sub-Gaussian and sub-exponential characterization [938, 1683]. To describe this result, we start with the formal definition of a sub-Weibull random variable:

Definition 29.1 (Sub-Weibull random variable, Vladimirova et al. [1683]). *A random variable X satisfying for all $t > 0$ and for some $C, \eta > 0$*

$$\mathbb{P}(|X| \geq t) \leq C \exp\left(-t^{1/\eta}\right), \quad (29.5)$$

is called a sub-Weibull random variable with tail parameter η , which is denoted by $X \sim \text{subW}(\eta)$.

Sub-Weibull distributions are characterised by tails lighter than (or equally light as) Weibull distributions; in the same way as sub-Gaussian or sub-exponential distributions correspond to distributions with tails lighter than Gaussian and exponential distributions, respectively. Sub-Weibull distributions are parametrised by a positive tail index η and are equivalent to sub-Gaussian for $\eta = 1/2$ and sub-exponential for $\eta = 1$.

Under a mild assumption on the activation function, namely that it scales roughly linearly as x shoots off to infinity, a notion made precise in Definition 29.2 in Section 29.5.1, we have the following theorem.

Theorem 29.3 (Theorem 3.1, Vladimirova et al. [1682]). *Consider a Bayesian neural network with $L - 1$ hidden layers, centred Gaussian priors and with activation function satisfying the extended envelope property of Definition 29.2. Then conditional on the input x , the marginal prior distribution on any unit (pre- or post-activation) of the l -th hidden layer is sub-Weibull with tail parameter $\eta = l/2$. That is for any $1 \leq l \leq L$, and for any $1 \leq m \leq H_l$,*

$$u_m^{(l)} \sim \text{subW}(l/2),$$

where $u_m^{(l)}$ is either a pre-activation $g_m^{(l)}$ or a post-activation $h_m^{(l)}$.

Thus, under certain conditions on the activation function (essentially, that it is ReLU-like and unbounded above), a Gaussian prior on weights induces a sub-Weibull($l/2$) distribution on the l -th layer units, where the tail parameter $l/2$ increases with l (larger tail parameter means heavier tail). For example, the first layer is Gaussian, the second layer is sub-exponential. Let us explain intuitively why this happens: multiplying many random matrices (even if each has nice properties) tends to produce outputs dominated by a few large random terms (due to ReLU passing through large positive sums), hence yielding fatter tails. A practical implication of this is that prior predictive distributions of a deep BNN can have high kurtosis: most of the time outputs look moderate, but occasionally they might be very large in magnitude due to a rare combination of large weights and reinforcing nonlinearities. This is a form of prior belief: namely, a deep random network prior believes that extreme activations, while rare, are more likely than a Gaussian belief would suggest. It is a departure from the infinite-width GP, which would predict output distributions to be Gaussian (no excess kurtosis).

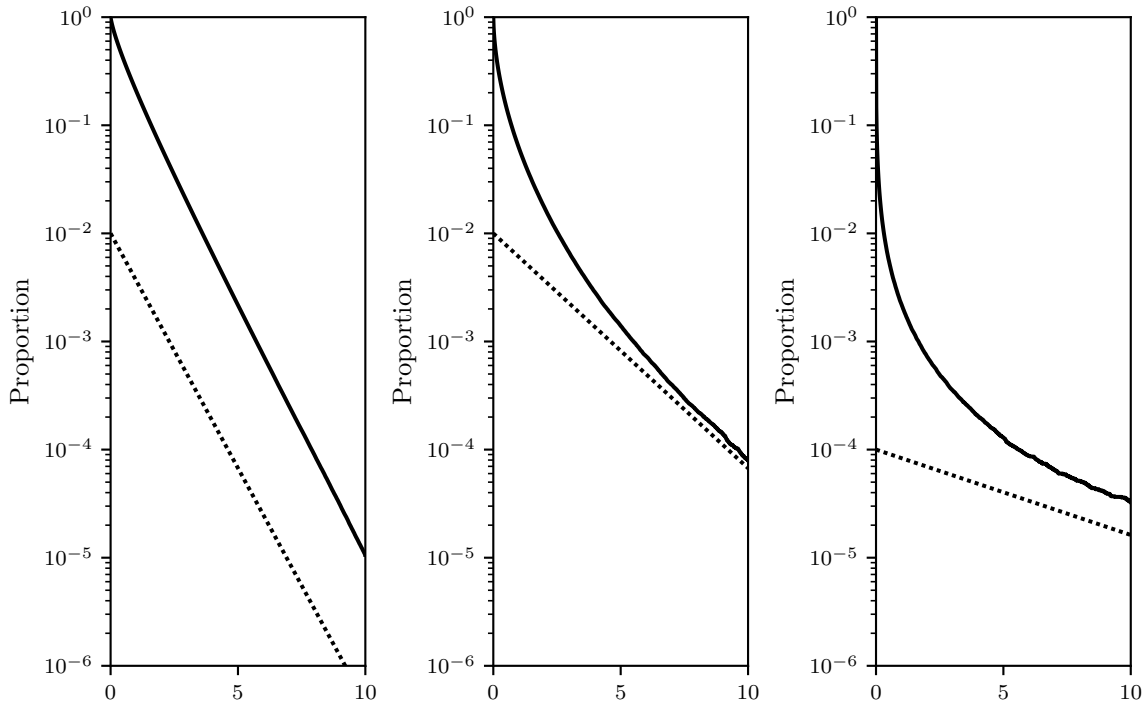


Figure 29.3: Verification of the node distribution. The plots show the tail behaviour for a node after 1 layer (left), 3 layers (middle), and 10 layers (right), both the empirical survival function (solid line) and the theoretical Weibull tail (dashed) are shown.

29.3.2 Meijer G -functions characterisation

Here, we introduce a precise characterisation of hidden units priors proposed by Noci et al. [1246], Zavatone-Veth and Pehlevan [1832] via so-called Meijer G -functions. These are very general special functions that encompass many classical functions (including hypergeometric and Bessel functions) as special cases. They are particularly useful because products of independent random variables often admit closed-form densities expressed through

Meijer G -functions Springer and Thompson [1548]. While these special functions are not commonly encountered in machine learning, they provide a compact way of expressing otherwise intractable densities and enable the derivation of analytical properties such as moments, tail behaviour, and asymptotic expansions.

With this property, Zavatore-Veth and Pehlevan [1832] and Noci et al. [1246] independently derived the density of the joint prior of the L -th layer $\mathbf{h}_L$ of a linear Bayesian neural network with Gaussian priors, with widths $H_1, \dots, H_L$, as

$$p_L^{\text{lin}}(\mathbf{h}_L | x) = \frac{\gamma_L}{(2^L \pi \kappa_L^2)^{H_L/2}} G_{0,L}^{L,0} \left(\frac{\|\mathbf{h}_L\|^2}{2^L \kappa_L^2} \middle| 0, (H_1 - H_L)/2, \dots, (H_{L-1} - H_L)/2 \right), \quad (29.6)$$

where

$$\kappa_L \equiv \sigma_1 \cdots \sigma_L \|x\| \quad \text{and} \quad \gamma_L \equiv \prod_{l=1}^{L-1} \frac{1}{\Gamma(H_l/2)}. \quad (29.7)$$

Here, $z \mapsto G_{0,L}^{L,0} \left(z \middle| 0, (H_1 - H_L)/2, \dots, (H_{L-1} - H_L)/2 \right)$ denotes a Meijer- G function with parameters specified by Zavatore-Veth and Pehlevan [1832]. Similarly, the authors show that the density of a ReLU Bayesian neural network has the following mixture form

$$p_L^{\text{ReLU}}(\mathbf{h}_L | x) = \left(1 - \frac{(2^{H_1} - 1)(2^{H_2} - 1) \cdots (2^{H_{L-1}} - 1)}{2^{H_1 + \cdots + H_{L-1}}} \right) \delta(\mathbf{h}_L) + \frac{1}{2^{H_1 + \cdots + H_{L-1}}} \sum_{k_1=1}^{H_1} \cdots \sum_{k_{L-1}=1}^{H_{L-1}} \binom{H_1}{k_1} \cdots \binom{H_{L-1}}{k_{L-1}} p_L^{\text{lin}}(\mathbf{h}_L | x; k_1, \dots, k_{L-1}, H_L), \quad (29.8)$$

where $\delta(\mathbf{h}_L)$ is the H_L -dimensional Dirac distribution, and where $p_L^{\text{lin}}(\mathbf{h}_L | x; k_1, \dots, k_{L-1}, H_L)$ denotes the linear BNN distribution of Equation (29.6) with specified hidden layer widths $(k_1, \dots, k_{L-1}, H_L)$. Using these exact representations, Zavatore-Veth and Pehlevan [1832] derive closed-form expressions for moments, establish tail bounds, and characterize the asymptotic behaviour of deep-layer priors. These results provide theoretical insight into how depth and width shape the distributions induced by Bayesian neural network priors.

29.3.3 Gaussian process limit for deep neural networks

The foundational result 29.2 has been generalised to more than one layer by de G. Matthews et al. [386], which we recall here.

Theorem 29.4 (Theorem 4, de G. Matthews et al. [386]). *Consider a Bayesian neural network with a continuous activation function satisfying the extended envelope property of Definition 29.2. Let each weight of the l -th layer follow independently a $\mathcal{N}(0, \sigma^2/H_l)$ distribution. Then the output of the network converges in distribution to a zero-mean Gaussian process as all layers widths $H_1, \dots, H_L$ tend simultaneously to ∞ .*

This theorem motivates the use of BNNs in settings where the function space prior is assumed to be a Gaussian process. This approach offers computational advantages in settings where a large number of point evaluations make direct manipulation of the GP kernel matrix infeasible, see also Chapter 26.

The above theorem is appealing in its generality, but does not give explicit expressions for the mean and covariance kernel of the resulting GP. For specific choices of activation functions and network architectures, it is possible to explicitly obtain the resulting GP mean function and covariance kernel, see e.g. Dutordoir et al. [477], Yang and Salman [1796]. Practitioners may be equally interested in the reverse situation, where one aims to construct a BNN prior based on a target GP on the output function space, one such construction is the Ridgelet prior by Matsubara et al. [1145], which results in a prior that is proven to converge to the desired GP as the layer width is taken to infinity.

Notably, Theorem 29.4 requires the variance of the prior to be scaled exactly proportional to σ^2/H_l with a *fixed* constant σ^2 . One may alternatively place a hyperprior on σ^2 , a setting which has been investigated by Caporali et al. [232] which shows that the resulting posterior converges to a Student- t process, as is made precise in the following theorem.

Theorem 29.5 (Corollary 3.8, Caporali et al. [232]). *The posterior of a 1-dimensional-output BNN ($H_L = 1$) with Gaussian-Inverse-Gamma prior and Gaussian likelihood, evaluated in the input x , converges in law to a multivariate Student- t variable as all layers widths $H_1, \dots, H_{L-1}$ tend simultaneously to ∞ .*

29.4 Different priors

Several priors were introduced in Chapter 28: informative priors, such as the spike-and-slab prior, which embed expert knowledge; uninformative and reference priors, aiming to minimise subjective influence; Gaussian priors for L_2 regularisation, controlling model complexity via variance; sparsity-inducing priors such as Laplace, scale-mixture, and horseshoe distributions [622, 623], encouraging simpler models; and model priors, enabling Bayesian model selection and averaging through structured weight inclusion mechanisms. Properties of the spike-and-slab prior and its role on posterior contraction rates are detailed in Section 29.6.2, following work by Jantre et al. [833], Polson and Ročková [1335]. We present two classes of priors in more details in the rest of the section: α -stable priors and trace-class priors.

29.4.1 α -stable priors

The family of α -stable priors were alluded to by Neal [1226] as candidates for avoiding the Gaussian process limit associated to wide neural networks. The α -stable distribution is characterised by a stability parameter $\alpha \in (0, 2]$, which controls the tail behaviour and the existence of moments. These distributions are defined by their characteristic function, as they typically lack closed-form densities (except for special cases like Gaussian, Cauchy, and Lévy). For a symmetric, centred α -stable random variable X , the characteristic function is:

$$t \mapsto \mathbb{E}[e^{itX}] = \exp(-\sigma^\alpha |t|^\alpha), \quad (29.9)$$

where $\sigma > 0$ is the scale parameter. When $\alpha = 2$, this reduces to a Gaussian distribution. When $\alpha < 2$, the distribution has heavy tails and infinite variance.

[1311] extends the classical Gaussian process limit of deep neural networks by replacing Gaussian weights with symmetric stable distributions. They show that, under appropriate scaling, the outputs of fully connected DNNs converge in distribution to a stochastic process whose finite-dimensional marginals are multivariate stable distributions. This stable process generalises the GP limit stated in Theorem 29.2 and Theorem 29.4, preserving heavy-tailed behaviour in the infinite-width limit. The parameters of the limiting distribution are obtained recursively across layers, and the convergence is established jointly over inputs and layers. Favaro et al. [507] extends these results by establishing sup-norm convergence rates under both joint and sequential layer-width growth, and formalising the recursive characterisation of the spectral measure that defines the limiting process.

29.4.2 Trace-class priors

While the infinite width limit of a standard BNN may be attractive at first glance, it results in unfortunate side effects noted by Neal [1227] in the discussion of his work: *“Infinite networks drawn from these priors do not have hidden units that represent ‘hidden features’ of the input. The ability to find such hidden features is often seen as an interesting aspect of neural network learning.”* One alternative to the uniform scaling of the prior variances is given by trace-class BNN priors [1488], which retain hidden units in the infinite width limit, and aid with identifiability as each weight has a distinct variance. The trace-class BNN prior on parameters θ consisting of weights $w_{i,j}^{(l)}$ and biases $b_i^{(l)}$ is given by the following independent Gaussian distributions

$$w_{i,j}^{(1)} \sim \mathcal{N}\left(0, \frac{\sigma_{w^{(1)}}^2}{i^\alpha}\right), \quad w_{i,j}^{(l)} \sim \mathcal{N}\left(0, \frac{\sigma_{w^{(l)}}^2}{(ij)^\alpha}\right) \text{ for } l \geq 2, \quad b_i^{(l)} \sim \mathcal{N}\left(0, \frac{\sigma_{b^{(l)}}^2}{i^\alpha}\right), \quad (29.10)$$

where $\alpha > 0$, $\sigma_{w^{(l)}}^2 > 0$ and $\sigma_{b^{(l)}}^2 > 0$ are variance-related parameters to be specified.

We now follow Sell and Singh [1488] and define the infinite width limit of the network. Given an infinite sequence of weights and biases for the first layer, distributed according to the prior of Equation (29.10), i.e. $\{(b_i^{(1)}, w_{i,1}^{(1)}, \dots, w_{i,D}^{(1)}) : i \in \mathbb{N}\}$, all the functions of the first layer, $\{g_i^{(1)} : i \in \mathbb{N}\}$, are clearly well defined. We define all the functions of the second layer corresponding to an infinite-width first layer to be the following almost sure limits, assuming they exist:

$$G_i^{(2)}(x) = \lim_{H_1 \rightarrow \infty} b_i^{(2)} + \sum_{j=1}^{H_1} w_{i,j}^{(2)} \phi(g_j^{(1)}(x)). \quad (29.11)$$

Assuming the random functions $\{G_i^{(2)} : i \in \mathbb{N}\}$ are well defined, which will be the case if $\alpha > 1/2$ as we show in Theorem 29.6, proceeding iteratively, all the functions $\{G_i^{(l)} : i \in \mathbb{N}\}$ of subsequent layers, $l = 3, \dots, L$, are

also well defined. The functions in each layer of a finite width network are denoted with lower case to clearly distinguish them from their infinite width versions. For the output layer, the finite network gives $g^{(L)}(x)$ while the infinite network gives $G^{(L)}(x)$.

A non-centred trace-class prior can be defined but requires the mean parameters to be appropriately summable. The σ parameters are overall scaling parameters of the priors variance, while α controls the decay of the weights in the width: large values result in quickly decaying weights in each layer, while small values result in a slower decay. The constraint $\alpha > 1/2$ ensures that the prior is well-defined in the infinite width limit as shown in the theorem below.

Theorem 29.6 (Sell and Singh [1488]). *Assume the activation function ϕ is 1-Lipschitz and satisfies $\phi(0) = 0$. Let $L \in \mathbb{N}$ and $\alpha > 0$. Fix further $\sigma_{w^{(l)}}^2 > 0$ and $\sigma_{b^{(l)}}^2 > 0$ for each $l \in [L]$. Then, the functions of the layers of the finite-width neural network satisfy*

1. *For all $x \in [0, 1]^D$ one has $|g_i^{(l)}(x)| < \infty$ almost surely, and $\mathbb{E}[g_i^{(l)}(x)] = 0$. Furthermore, there exists σ_i^2 such that $\mathbb{E}[(g_i^{(l)}(x))^2] < \sigma_i^2/i^\alpha$ for all $x \in [0, 1]^D$. Here, both expectations are taken with respect to the prior on the parameters θ of the neural network. In particular this property holds for $g_1^{(L)}(x)$.*
2. *There exists $c_l \geq 0$ such that for all $x_1, x_2 \in [0, 1]^D$, $\mathbb{E}\left[\left(g_i^{(l)}(x_1) - g_i^{(l)}(x_2)\right)^2\right] \leq c_l \|x_1 - x_2\|^2 / i^\alpha$, with the expectation again taken with respect to the prior. In particular this gives $\mathbb{E}\left[\left(g_1^{(L)}(x_1) - g_1^{(L)}(x_2)\right)^2\right] \leq c_L \|x_1 - x_2\|^2$.*

In addition, if $\alpha > 1/2$, the functions on every layer of the infinite-width neural network, see Equation (29.11), exist almost surely and satisfy the above Properties 1 and 2, when the functions $g_i^{(l)}(x)$ therein are replaced with $G_i^{(l)}(x)$ as defined in Equation (29.11).

The trace-class prior is attractive due to its interpretability of the functions on the last hidden layers, which do not converge to 0 as one considers the infinite-width limit. It further improves model identifiability, as swapping nodes changes the prior probabilities for the resulting network. The natural ordering of the respective weights lends the trace-class prior as an obvious choice for partial or subnetwork inference [242], as well as a valuable prior for multilevel Monte Carlo, see Chapter 4. The hyperparameters can either be chosen by the user, or inferred by placing a hyperprior on each of them.

29.5 Different architectures

29.5.1 Different activation functions

As described in previous sections, the forward propagation of priors on weights induces progressively heavier-tailed distributions on the units. In this section, we examine how these properties interact with different activation functions. To this aim, the activation function ϕ is required to span at least half of the real line as follows. We introduce an extended version of the activation function assumption from Matthews et al. [386]:

Definition 29.2 (Extended envelope property, Vladimirova et al. [1684]). *An activation function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is said to obey the extended envelope property if there exist $c_1, c_2 \geq 0$, $d_1, d_2 > 0$ such that the following inequalities hold*

$$|\phi(u)| \geq c_1 + d_1|u| \quad \text{for all } u \in \mathbb{R}_+ \text{ or } u \in \mathbb{R}_-, \quad (29.12)$$

$$|\phi(u)| \leq c_2 + d_2|u| \quad \text{for all } u \in \mathbb{R}. \quad (29.13)$$

The interpretation of this property is that ϕ must shoot to infinity at least in one direction ($\mathbb{R}_+$ or $\mathbb{R}_-$, at least linearly (29.12), and also at most linearly (29.13). Of course, compactly supported activation functions such as sigmoid and tanh do not satisfy the extended envelope property but the majority of other activation functions do, including ReLU, ELU, PReLU, and SeLU.

29.5.2 Different layers

While fully connected networks have been the focus of much of the preceding analysis, modern architectures frequently incorporate structural innovations such as residual connections, convolutional layers, and continuous-depth models. These modifications significantly affect the prior and its induced function space behaviour.

Residual layers. Residual connections, as introduced in ResNets, can be viewed as introducing an identity mapping in addition to the learned transformation [721]. Formally, the pre-activation at layer l , $g^{(l)}$, can be written as:

$$g^{(l)} = g^{(l-1)} + r^{(l)}(g^{(l-1)}; \theta^{(l)}), \quad (29.14)$$

for some mapping $r^{(l)}$. Under Bayesian treatment, each residual block contributes a random perturbation to the input of the next layer. This modifies the prior by inducing a random walk structure over functions, leading to smoother transitions across layers. The residual architecture results in slower growth of variance with depth compared to standard feed-forward architectures, and may suppress heavy-tailed behaviour [1684].

Convolutional layers. Convolutional networks apply shared filters across spatial locations [993]. Let $\theta^{(l)}$ be a convolutional kernel and $*$ denote the convolution operator. Then the pre-activation at layer l , $g^{(l)}$, is given by:

$$g^{(l)} = \theta^{(l)} * h^{(l-1)} + b^{(l)}, \quad (29.15)$$

where $h^{(l-1)}$ denotes the post-activation at layer $l - 1$. Due to weight sharing and local receptive fields, the induced function-space prior has strong spatial correlations and invariances. This structure leads to a significant reduction in the number of free parameters and typically induces smoother and more data-adaptive priors, which can outperform corresponding infinite-width GPs or neural tangent kernels [829] in practical settings.

Neural ODEs. Neural ODEs [272] generalise discrete-depth networks by considering a continuous evolution of hidden states. For a hidden state $h(t)$ and parametrised vector field $v(h, t; \theta)$, the dynamics are governed by:

$$\frac{dh(t)}{dt} = v(h(t), t; \theta), \quad h(0) = x. \quad (29.16)$$

The output of the network is then given by solving the initial value problem over some interval $[0, T]$. The induced prior on functions arises through the distribution over θ :

$$h(T) = x + \int_0^T v(h(t), t; \theta) dt. \quad (29.17)$$

If v is linear in h and $\theta \sim \mathcal{N}(0, I)$, then $h(t)$ behaves similarly to a Gaussian process in time [574]. More generally, the prior is implicit and can be seen as a stochastic process in function space, influenced by the ODE solver and integration method. The continuous-time formulation enables adaptive computation and smoother transitions, but complicates prior predictive analysis, as closed-form density expressions are generally unavailable.

In practice, modern architectures often combine multiple structural motifs, e. g., residual connections embedded within convolutional layers or ODE blocks used in sequence models. This results in complex, hierarchical priors that defy simple analytic characterisation. These hybrid designs further emphasise the role of architectural choices in shaping unit priors.

29.6 Influence of the prior on the posterior

29.6.1 Bayesian asymptotics

The field of Bayesian asymptotics focuses on the behaviour of the posterior distribution $p(\theta \mid \mathcal{D}^n)$ as the sample size n of the data $\mathcal{D} = \mathcal{D}^n$ tends to infinity [617–619]. In settings where a true data-generating process characterised by a true parameter θ_0 is assumed to exist, the field of Bayesian asymptotics aims to establish the following guarantees: *posterior consistency* at θ_0 refers to $p(U^c \mid \mathcal{D}^n) \rightarrow 0$ as $n \rightarrow \infty$ in P_{θ_0} -probability for all neighbourhoods U of θ_0 , i. e. $p(\theta \mid \mathcal{D}^n)$ concentrates on the true θ_0 ; *contraction rates* describe how quickly $p(\theta \mid \mathcal{D}^n)$ concentrates around θ_0 , i. e. how quickly neighbourhoods U can shrink with n [617, 619]; *Bernstein–von Mises (BvM) theorems* are types of Central Limit theorems which state that under regularity conditions, $p(\theta \mid \mathcal{D}^n)$ converges to a Gaussian centred at the maximum likelihood estimator $\hat{\theta}^{\text{MLE}}$, with covariance given by the inverse Fisher information [175, 1547].

Bayesian neural networks allow us to place prior distributions over network weights and perform inference through the resulting posterior distribution. Understanding the impact of the prior on the asymptotic behaviour of the posterior is critical to establishing theoretical guarantees and bridging the gap between Bayesian learning and its frequentist counterpart. This section provides an overview of the asymptotic theory of BNNs, focusing on three key aspects: contraction rates, adaptation, and Bernstein–von Mises theorems. Many of the results below rely on regularity assumptions. Their counterparts for singular models, which frequently arise in Bayesian Deep Learning, are discussed in Chapter 52 on Singular Learning Theory.

29.6.2 Posterior contraction rates

Before stating the main result of posterior concentration for sparse ReLU networks with spike-and-slab prior [1335], we introduce the required notation. Let f_0 denote the unknown true regression function defined on $[0, 1]^D$, assumed to belong to a Hölder class of smoothness $\alpha > 0$. We observe data (Y_i, x_i) , where

$$Y_i = f_0(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, 1), \quad i = 1, \dots, n,$$

the restriction to $\mathcal{N}(0, 1)$ noise is for simplicity. We define a Bayesian neural network prior using deep ReLU networks of depth L , width N , and sparsity s , equipped with a spike-and-slab prior on the weights and offsets. The architecture parameters L , N , and s are chosen depending on the smoothness level α , the dimension D , and the sample size n , to achieve optimal approximation properties.

Denote by $\|\cdot\|_n$ the empirical L^2 norm on the training data, and let $\epsilon_n = n^{-\alpha/(2\alpha+D)} \log^\delta(n)$ for some $\delta > 1$ denote the near-minimax rate for estimating an α -smooth function in dimension D .

Theorem 29.7 (Posterior concentration, simplified from Theorem 6.1 of Polson and Ročková [1335]). *Assume that f_0 is α -Hölder smooth with $\alpha < D$ and uniformly bounded. Then, for a suitably chosen network architecture depending on α , D , and n , the posterior distribution concentrates around f_0 at the rate ϵ_n , i. e.,*

$$\Pi\left(f : \|f - f_0\|_n > M_n \epsilon_n \mid Y^{(n)}\right) \rightarrow 0$$

in probability as $n \rightarrow \infty$, for any sequence $M_n \rightarrow \infty$.

Significant progress has been made in establishing posterior contraction rates for BNNs in nonparametric settings. Fully connected deep neural networks have been shown to achieve (near) optimal rates in tasks such as regression [244, 911] and multiclass classification [190]. These results suggest that neural networks possess a remarkable capacity for automatic adaptation to unknown function smoothness.

While classical spike-and-slab priors, that are mixtures of a point mass and a diffuse slab, have been extensively used to achieve optimal rates [833, 1335], their computational complexity makes them less practical in deep learning contexts. Recent developments have focused on alternative priors that retain theoretical optimality while improving computational feasibility [916, 1006].

Contraction rates for variational approximations to BNN posteriors are explored by Bhattacharya and Maiti [150], Bhattacharya et al. [151], offering insights into the theoretical performance of scalable inference methods. In parallel, related results are derived for deep Gaussian process priors [361, 463], with recent advances by Bachoc and Lagnoux [72], Castillo and Randrianarisoa [245], Finocchio and Schmidt-Hieber [519] who characterise their convergence behaviour.

The high-dimensional nature of modern applications raises the question of how posterior rates scale with both sample size n and input dimension D . While most existing results assume a fixed D , Bhattacharya et al. [152] is a notable exception, deriving minimax-optimal rates that depend jointly on (n, D) , and thus addressing more realistic big data scenarios.

29.6.3 Adaptation

Beyond contraction rates, recent work has shown that overparameterised networks with heavy-tailed priors can achieve automatic statistical adaptation, with posterior distributions concentrating at minimax-optimal rates without prior knowledge of function smoothness [244]. This connects with earlier findings that Gaussian priors over network weights can induce increasingly heavy-tailed behaviour through the layers of a deep network [1682].

29.6.4 Bernstein–von Mises theorems

Bernstein–von Mises (BvM) theorems describe the asymptotic normality of posterior distributions, providing a Bayesian analogue to the classical central limit theorem. In the context of BNNs, exact BvM results are difficult to obtain owing to non-identifiability, symmetries and overparameterisation of the models.

For shallow neural networks, with moderate depth and width and well-behaved priors, classical BvM behaviour can be recovered. However, as networks become deeper and more expressive, posterior distributions deviate significantly from Gaussian behaviour, and standard BvM theorems fail to hold [175].

Current research aims to extend BvM-type results to functionals of the posterior, such as predictions or feature-specific summaries, rather than the entire parameter space. This more realistic approach aligns with the practical use of BNNs, where uncertainty over predictions is often of primary interest.

29.7 Discussion and open questions

Finite-width networks induce non-Gaussian priors over the output functions, but retain certain “Bayesian” properties. For instance, by de Finetti’s theorem [383], one can view the finite network’s function distribution as an exchangeable random process indexed by inputs, but exact distributional forms are intractable beyond very small networks or only under highly restrictive assumptions. The non-Gaussianity means finite networks can, in principle, capture more complex prior beliefs (e.g. multimodal function distributions), but it also complicates analysis. In contrast, in the infinite-width limit, many BNN priors yield a Gaussian process over the output functions, simplifying theoretical analyses and opening exciting connections to the study of GPs.

In some cases, finite-width neural networks outperform their infinite-width counterparts [49, 477, 587, 1001]. Arora et al. [49] show that convolutional neural networks outperform their corresponding limiting neural tangent kernel (NTK, see Section 8.2, and Jacot et al. [829]). This performance gap is likely due to the finite width effect where a fixed kernel cannot fully describe the convolutional neural network dynamics. The evolution of the NTK along with training has its benefits on generalisation as shown in further works [482, 796]. However, since the performance of finite and infinite-width neural networks is different, it is still not clear when the GP framework is more amenable to describe the BNN behaviour.

Chapter 30

Informative priors

This chapter introduces a framework for constructing informative, data-driven prior distributions over the parameters of neural networks [1436, 1437]. Specifying meaningful priors for neural-network models is inherently difficult because it is rarely obvious how to map a particular parameter configuration to a desired functional behaviour, such as good calibration, high predictive uncertainty under semantic shift (for example, on inputs from classes absent from the training data), a low predictive negative log-likelihood, or robustness to other distribution shifts.

The framework uses problem-specific information and auxiliary data to assign high prior density to parameter values that induce these desired functional properties. We discuss two specific instantiations:

1. **Uncertainty-aware priors**, designed to improve uncertainty quantification by favouring models that exhibit high predictive entropy on inputs meaningfully distinct from the training distribution (Section 30.2);
2. **Group-aware priors**, designed to improve group robustness by favouring models that generalise well to data subgroups underrepresented in the training data (Section 30.3).

30.1 A Family of Informative Data-Driven Priors

We begin by formalising the model structure. Let $\theta \in \Theta$ denote a realisation of the neural-network parameter vector, and let $p(y | x, \theta; f)$ be the observation model induced by the network f . For classification, f may output logits that parameterise a categorical distribution; for regression, it may parameterise, for example, the mean of a Gaussian observation model. We partition $\theta = (\theta_h, \theta_L)$ into the final-layer parameters θ_L and the parameters θ_h of all preceding layers (the feature extractor). The network mapping is thus $f(\cdot; \theta) := h(\cdot; \theta_h)\theta_L$, where $h(\cdot; \theta_h)$ is the post-activation output of the penultimate layer. We assume access to pre-trained feature-extractor parameters θ_h^* and to context data encoding useful information about the downstream task.

Our framework is built on a *context inference problem*. We use a batch of context inputs $x_c = \{x_1, \dots, x_M\}$ and corresponding labels $y_c = \{y_1, \dots, y_M\}$, drawn from a joint distribution p_{X_c, Y_c} . Capital letters X_c and Y_c denote random variables, whereas x_c and y_c denote their realisations. We then introduce a Bernoulli random variable $\tilde{Z} \in \{0, 1\}$ indicating whether θ induces a model with a desired property; for example, $\tilde{Z} = 1$ may indicate high uncertainty on x_c or high accuracy on evaluation points with a specified group attribute.

We express this relationship through the context observation model $\check{p}(\tilde{z} | \theta; f, p_{X_c, Y_c})$, the probability mass function of the realisation $\tilde{z}$ conditional on θ . The quantities following the semicolon are fixed ingredients used to construct the model, not additional random conditioning variables. Given a base prior $p(\theta)$, the informative data-driven prior is the posterior distribution over θ conditional on the desired outcome $\tilde{z} = 1$. For notational simplicity, we suppress fixed arguments such as f where no ambiguity arises. By Bayes' theorem,

$$p(\theta | \tilde{z} = 1; p_{X_c, Y_c}) = \frac{\check{p}(\tilde{z} = 1 | \theta; p_{X_c, Y_c})p(\theta)}{\check{p}(\tilde{z} = 1; p_{X_c, Y_c})}. \quad (30.1)$$

This conditional distribution is then used as the prior in a subsequent inference problem for the neural-network parameters, for example in maximum a posteriori (MAP) estimation or in approximate Bayesian inference such as mean-field variational inference (Chapter 13).

To specify the context observation model, we interpret $\tilde{Z} = 1$ as achieving the desired property (e.g., reliable

uncertainty). We link the probability of this outcome to a non-negative cost function c :

$$\begin{aligned}\tilde{p}(\tilde{z} = 1 \mid \theta; p_{X_c, Y_c}) &:= \exp(-\mathbb{E}_{p_{X_c, Y_c}}[c(X_c, Y_c, \theta)]), \\ \tilde{p}(\tilde{z} = 0 \mid \theta; p_{X_c, Y_c}) &:= 1 - \tilde{p}(\tilde{z} = 1 \mid \theta; p_{X_c, Y_c}).\end{aligned}\quad (30.2)$$

Because c is non-negative, these two quantities define a valid Bernoulli probability mass function. Equations (30.1) and (30.2) show that the usefulness of the resulting prior depends on the cost function, context distribution, and base prior. The principal modelling challenge is therefore to choose c and p_{X_c, Y_c} so that the context observation model captures the intended functional property as faithfully as possible.

30.2 Uncertainty-Aware Data-Driven Priors

Following Rudner et al. [1438], we now instantiate this framework to construct an uncertainty-aware prior for fine-tuning pre-trained classifiers.

Base prior. We specify layer-specific base priors. For the feature-extractor parameters θ_h , we use a Gaussian prior centred on the pre-trained parameters: $p(\theta_h) = \mathcal{N}(\theta_h \mid \theta_h^*, \tau_h^{-1} \mathbf{I})$. This encourages the solution to remain close to the pre-trained representation. For the final-layer parameters, we use a standard zero-mean Gaussian prior: $p(\theta_L) = \mathcal{N}(\theta_L \mid \mathbf{0}, \tau_L^{-1} \mathbf{I})$.

Cost function and context distribution. For this classification construction, $f(x_c; \theta)$ denotes the logits evaluated on the M context inputs. For each output coordinate k , $[f(x_c; \theta)]_k \in \mathbb{R}^M$ collects the corresponding logits across the context batch. We define the cost using a squared Mahalanobis distance between these logits and a target mean:

$$c(x_c, y_c, \theta) := \tau_c \sum_{k=1}^K D_{\mathcal{M}}^2([f(x_c; \theta)]_k, m(x_c, y_c)_k, C(x_c)), \quad (30.3)$$

where $\tau_c > 0$ is a cost-scaling parameter, distinct from the base-prior precisions τ_h and τ_L , and K is the number of output dimensions. The squared Mahalanobis distance in Equation (30.3) is

$$D_{\mathcal{M}}^2(\mathbf{a}, \mathbf{b}, \mathbf{C}) := (\mathbf{a} - \mathbf{b})^\top \mathbf{C}^{-1} (\mathbf{a} - \mathbf{b}). \quad (30.4)$$

To encourage high uncertainty on the context inputs, we set $m(x_c, y_c)_k := [y_c]_k$ and choose the context-label distribution to be a point mass at zero, $p(y_c \mid x_c) := \delta_{\mathbf{0}}(y_c)$. This makes the target mean zero almost surely under the context-label distribution.

We define the covariance function by

$$C(\cdot) := s_1 h(\cdot; \theta_h^*) h(\cdot; \theta_h^*)^\top + s_2 \mathbf{I}, \quad (30.5)$$

where $s_1 > 0$ and $s_2 > 0$ are fixed scaling parameters. The form in Equation (30.5) reflects the structure of the pre-trained model's representations.

Finally, we take $p(x_c)$ to be an empirical distribution over inputs meaningfully distinct from the training set, such as out-of-distribution inputs. By Equations (30.2) and (30.3), a low cost—and hence a high value of $\tilde{p}(\tilde{z} = 1 \mid \theta)$ —is obtained when the logits are close to the zero target in the Mahalanobis metric of Equation (30.4). For a softmax classifier, logits close to zero produce a near-uniform predictive distribution, so the resulting prior favours high predictive uncertainty on the specified out-of-distribution context inputs.

30.3 Group-Aware Data-Driven Priors

Following Rudner et al. [1437], we specify a useful group-aware prior by choosing (i) a context distribution p_{X_c, Y_c} for which observing $\tilde{z} = 1$ is informative and (ii) a cost function c that serves as a proxy for group robustness under that distribution.

Both choices are generally challenging. One could, for example, learn a tailored generative model for the context distribution or construct a more elaborate cost function. Here we use a fixed, pre-specified context distribution and a simple cost function to illustrate the framework.

Context distribution. We assume access to a small validation dataset $\mathcal{D}_{\text{val}}$ containing group information. We form the context distribution by upweighting rare groups in this dataset:

$$p_{X_c, Y_c} = \sum_{g \in \mathcal{G}} \alpha_g p_g, \quad \alpha_g = \frac{\tilde{\alpha}_g}{\sum_{g' \in \mathcal{G}} \tilde{\alpha}_{g'}}, \quad (30.6)$$

$$\tilde{\alpha}_g = \left(\frac{|\mathcal{D}_{\text{val}}|}{|\mathcal{D}_{\text{val}}^g|} \right)^\gamma.$$

Here p_g is the empirical distribution of validation examples in group g , $|\mathcal{D}_{\text{val}}^g|$ is the number of such examples, and $\gamma \geq 1$ controls the strength of the upweighting. The construction does not require the validation-group frequencies to match the training-group frequencies; it deliberately defines a reweighted context distribution. As Equation (30.6) shows, increasing γ increases the relative emphasis on rarer groups.

Cost function. We define a cost that acts as a proxy for group robustness, inspired by sharpness-aware minimisation (SAM) [544]. The task loss $\ell(\cdot)$ —typically cross-entropy for classification or mean-squared error for regression—is evaluated at a first-order worst-case parameter perturbation:

$$c(x_c, y_c, \theta) := \ell(y_c, f(x_c; \theta + \rho \epsilon(\theta, x_c, y_c))), \quad (30.7)$$

where $\rho > 0$ is the perturbation magnitude. For a non-zero loss gradient, the perturbation direction is

$$\epsilon(\theta, x_c, y_c) := \perp \left(\frac{\nabla_\theta \ell(y_c, f(x_c; \theta))}{\|\nabla_\theta \ell(y_c, f(x_c; \theta))\|_2} \right), \quad (30.8)$$

where $\perp(u)$ denotes the **stop_gradient** operator: it has forward value u but derivative zero. When the loss gradient vanishes, we set $\epsilon(\theta, x_c, y_c) = \mathbf{0}$. The normalised gradient is the unit- ℓ_2 direction that maximises the first-order increase in the loss; the stop-gradient operator treats this direction as fixed when differentiating c .

Low loss is generally associated with good generalisation, but the small validation set contains few examples from rare groups, making generalisation to those groups difficult. The context distribution counteracts this imbalance by upweighting rare-group examples, while the perturbed loss favours parameter configurations that remain low-loss under a local worst-case perturbation. Such configurations are expected to generalise better to the upweighted rare-group context data.

The stop-gradient operator changes the derivative used in optimisation without changing the forward value of the perturbation. To make this explicit, let $\tilde{\epsilon}(\theta, x_c, y_c)$ denote the same normalised gradient as in Equation (30.8), but without stop-gradient, and let θ_t denote the current parameter iterate. Using column-vector gradients, holding x_c and y_c fixed, writing $J_{\tilde{\epsilon}}(\theta_t)$ for the Jacobian of $\tilde{\epsilon}$ with respect to θ at θ_t , and using ϑ as a dummy parameter vector, the chain rule gives

$$\begin{aligned} & \nabla_\theta \ell(y_c, f(x_c; \theta + \rho \tilde{\epsilon}(\theta, x_c, y_c)))|_{\theta=\theta_t} \\ &= (\mathbf{I} + \rho J_{\tilde{\epsilon}}(\theta_t))^\top \nabla_\vartheta \ell(y_c, f(x_c; \vartheta))|_{\vartheta=\theta_t + \rho \tilde{\epsilon}(\theta_t, x_c, y_c)}. \end{aligned} \quad (30.9)$$

With stop-gradient, automatic differentiation omits the Jacobian contribution in Equation (30.9) and returns

$$g_{\text{sg}}(\theta_t) := \nabla_\vartheta \ell(y_c, f(x_c; \vartheta))|_{\vartheta=\theta_t + \rho \epsilon(\theta_t, x_c, y_c)}. \quad (30.10)$$

Thus, the forward value in Equation (30.7) still defines the cost used in the context observation model, but g_{sg} is a SAM-style surrogate gradient rather than the exact gradient of that forward scalar cost. Without additional integrability conditions, this surrogate vector field need not be the gradient of a globally defined alternative scalar potential.

30.4 Summary

The framework in this chapter constructs informative priors by conditioning a base prior on the event that the induced model exhibits a desired functional property. The uncertainty-aware construction combines out-of-distribution context inputs with a Mahalanobis cost that favours near-uniform predictions, whereas the group-aware construction combines an upweighted rare-group validation distribution with a SAM-style perturbed loss. In both cases, the quality of the prior depends on how faithfully the context distribution and cost encode the intended property; developing broader classes of well-specified context observation models remains an open direction.

Chapter 31

Function-space priors

As discussed in Chapter 28, in Bayesian statistics, the choice of prior distributions over model parameters represents a fundamental part of the modelling process [601]. In many applications, the elicitation of such prior information on model parameters can be attempted through domain expertise (see, e.g., Lindley et al. [1051], O’Hagan [1255]).

However, for deep learning models, this is not straightforward, and this is the focus of this chapter. Without loss of generality, we are going to consider the case of fully connected Neural Network (NN) models; much of the discussion applies to different architectures too. Let $f(x, \theta)$ denote the output obtained by a NN with parameters θ for an input x . There are a number of challenges in choosing sensible priors for θ due to the fact that NNs are typically over-parameterised. Over-parameterisation in NNs means that the number of parameters can be very large (easily beyond a few millions) leading to statistical non-identifiability, and requiring one to pick a prior for every single parameter is a daunting task. Blocking strategies can alleviate this burden, e.g., by assigning the same prior to parameters in the same layer. However, a fundamental problem still remains: *what is the effect produced by the choice of a particular prior distribution over θ on the prior distribution over $f(x, \theta)$?*

This chapter is dedicated to the study of such prior distributions over $f(x, \theta)$. In Section 31.1, we provide some background on function-space priors defined as the priors over $f(x, \theta)$ induced by the choice of a prior over θ . After discussing the intractability of characterising these priors analytically in Section 31.1.1 and Section 31.1.2, we focus on ways in which we can obtain samples from these priors in Section 31.1.4. We then move on to discussing in Section 31.2 how the choice of convenient priors may lead to pathological priors in function-space. Indeed, astute readers with background in Bayesian Statistics may find it easy to circumvent the problem of specifying priors over NN parameters by opting for weakly informative priors over θ . Unfortunately, such a naïve and seemingly benign approach may lead to function-space priors over $f(x, \theta)$ with undesirable properties; see Tran et al. [1627] for a study of Bayesian model selection in the context of unsupervised learning problems.

We conclude this chapter with Section 31.3 where we study ways in which we can choose priors over θ inducing function-space priors exhibiting desirable properties. The majority of the focus of this latter section is on matching NN function-space priors with function-space priors of interest and for which we have easy access to samples. This is realised by casting this matching task as the optimisation of a suitable objective, and we present a number of options by drawing from the literature on divergences and integral probability metrics. All in all, this chapter should convince readers that care needs to be taken when choosing priors for Bayesian deep learning, and that there are practical strategies to adopt in order to do so.

31.1 From priors over parameters to priors over functions

For simplicity of exposition, let’s focus on the case of a supervised learning problem where the aim is to establish a mapping between inputs $x \in \mathcal{X}$, with $\mathcal{X} \subset \mathbb{R}^D$, and labels $y \in \mathcal{Y}$ given a collection of pairs (x_i, y_i) with $i = 1, \dots, N$. Consider the following definition of a NN-based model with parameters θ :

$$f(x, \theta) \in \mathbb{R}^C, \quad \text{with} \quad \theta \in \mathbb{R}^P, \quad (31.1)$$

which we can use, together with an observation model, to define a likelihood function $p(y_1, \dots, y_N \mid x_1, \dots, x_N, \theta)$ measuring the goodness of fit of the model to the given data (equivalently, one may write $p(y_{1:N} \mid x_{1:N}, \theta; f)$ to make the dependence on the network map explicit). For now we do not write a specific form for the likelihood, as it is not the focus of the discussion. In order to complete the specification of the Bayesian NN model for the given data, we need to choose a prior for θ :

$$\theta \sim p(\theta). \quad (31.2)$$

With a complete model specification, we can easily realise that the posterior over θ is intractable and that we need approximations. We will come back to this important point later in the chapter, but for now let's focus our discussion on how to choose $p(\theta)$ sensibly.

The choice of a prior over θ induces a distribution over $f(x, \theta)$, meaning that it yields a prior distribution over the NN output for any given input x . More formally, let's denote by $F_{1:n} = (F_1, \dots, F_n)$ with $F_i \in \mathbb{R}^C$ the collection of random variables representing the NN function associated with the inputs $x_1, \dots, x_n$. Note that the inputs are not necessarily the training inputs here, so we denote the number of such inputs as n instead of N . We can obtain the prior distribution over $F_{1:n}$ by marginalising out θ as follows:

$$p(F_{1:n}) = \int p(F_{1:n} | \theta) p(\theta) d\theta = \int \delta(F_{1:n} - (f(x_1, \theta), \dots, f(x_n, \theta))) p(\theta) d\theta. \quad (31.3)$$

The function $\delta(\cdot)$ represents the Dirac delta, and we use it to highlight the deterministic nature of the forward computations of the NN which gives the components $f(x_i, \theta)$ for a given input x_i and θ .

31.1.1 Function-space priors are desirable but intractable

Equation 31.3 expresses the distribution over the output of the NN. Why is this of any interest? And what kind of properties should we expect this distribution to have? Earlier, we discussed how in Bayesian inference it is important to include prior knowledge from experts into a modelling problem and how this principle is of little help in guiding the choice of a prior over NN parameters θ . In contrast, it is often much easier to reason about the kind of functions we may want to model *a priori*. In other words, function-space priors are more “interpretable”; it is much easier to reason about macroscopic properties that we would like to encode in the functions represented by NNs, such as, e.g., degree of smoothness, periodicity, and marginal variance, rather than dealing with individual NN parameters. Indeed, the statistical nonidentifiability and nonlinearity induced by the over-parameterisation of NNs make it challenging to study the sensitivity of the function-space prior with respect to the prior over individual parameters.

Unfortunately, the integral in Equation 31.3 is generally intractable, again due to the nonidentifiability and nonlinearity of the NN parameterisation, and reasoning about their function-space priors is not immediately possible. Considering a Gaussian prior $p(\theta)$, for example, it is easy to see that the nonlinear dependence of $f(x, \theta)$ from θ makes it intractable to calculate this integral. In Section 31.3 we are going to discuss strategies to determine prior distributions over θ so that function-space priors exhibit properties of interest. Before doing so, we present more insights into the function-space prior induced by a prior over the parameters by viewing Equation 31.3 as a change of variables operation and study it in the context of neural processes.

31.1.2 Interpretation of function-space priors as a change of variables

Another way to interpret the operation carried out in Equation 31.3 is to view it as a change of variables, where the prior measure over θ is transformed into a measure over $F_{1:n}$. Here n represents the number of input locations at which we evaluate the function-space prior. Recalling that $x_i \in \mathcal{X}$ with $\mathcal{X} \subset \mathbb{R}^D$, imagining that we are interested in analysing the behaviour of the function $f(x, \theta)$ at the n input locations, we are then interested in the distribution of $F_{1:n}$. For reference, if T were a differentiable bijection between spaces of the same dimension and $F = T(\theta)$, the usual change-of-variables formula would give

$$p_F(f) = p_\theta(T^{-1}(f)) |\det J_{T^{-1}}(f)|.$$

The map induced by a neural network at fixed inputs, $T_{x_{1:n}}(\theta) = (f(x_1, \theta), \dots, f(x_n, \theta))$, rarely satisfies these assumptions: it is typically many-to-one and often maps parameter space into an output space of different dimension. It is clear that the mapping taking θ as input and mapping these into $F_{1:n}$ is not a bijection in general, which makes the Jacobian rank deficient (rank at most the minimum between P and nC).

This view, however, reveals something interesting about the properties that we can expect to observe in the prior over $F_{1:n}$. Assuming $nC > P$, the change of variables formula indicates that the prior over $F_{1:n}$ is going to live in a low-dimensional manifold of P dimensions. We may be tempted to still characterise the change of measure over this manifold by computing the determinant of the Jacobian through the computation of its top P eigenvalues; however, this requires computing and factorising the Jacobian, which is an $(nC) \times P$ matrix, requiring $\mathcal{O}(nC P^2) + \mathcal{O}(P^3)$ computations. Thus the change-of-variables view is useful mainly as a diagnostic: it explains why evaluating the function-space density is difficult and why the induced distribution may concentrate on a low-dimensional set. The practical methods below therefore rely on samples from $p(F_{1:n})$ rather than on pointwise evaluations of this density.

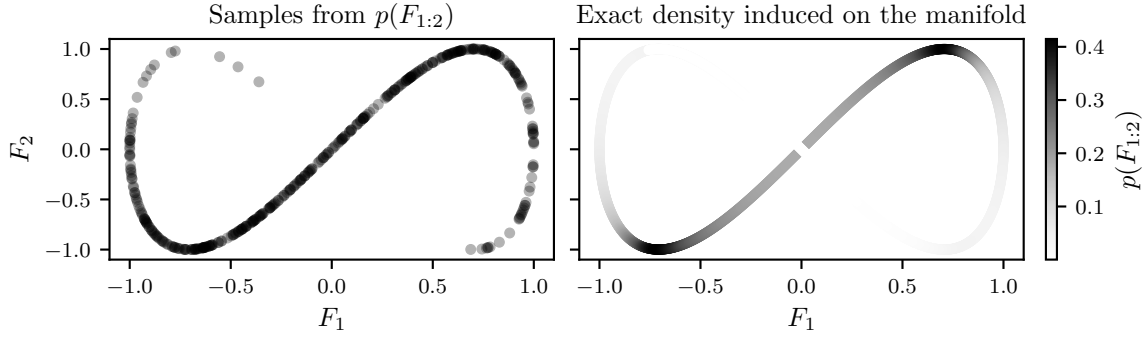


Figure 31.1: Visualisation of the mapping $(F_1, F_2) = (\sin(\theta x_1), \sin(\theta x_2))$ for $\theta \sim \mathcal{N}(0, 1)$ at input locations $x_1 = 1$ and $x_2 = 2$. (Left) Monte Carlo samples showing that all points lie on a one-dimensional sinusoidal curve in $[-1, 1]^2$; (Right) Exact induced density along this curve, computed via the density-on-manifold change-of-variables formula $p(F_{1:2}) = p(\theta) / \|dg(\theta)/d\theta\|_2$.

To visualise this phenomenon better, consider a single-parameter sine network evaluated at two input points, $x_1, x_2 \in \mathbb{R}$, that is $g : \theta \mapsto (F_1, F_2) = (\sin(\theta x_1), \sin(\theta x_2))$. Note that this is the case discussed before, where the number of input points ($n = 2$) is greater than the number of parameters ($P = 1$). Figure 31.1 shows what happens to $p(F_{1:2})$. When we sample $\theta \sim \mathcal{N}(0, 1)$ and plot the resulting pairs (F_1, F_2) in the plane, the points concentrate along a smooth, closed curve in $[-1, 1]^2$ rather than filling the square, as expected by the low-rank form of the Jacobian. In this simple case, we can also derive explicitly the induced density along the curve, using the change-of-variable formulas.

31.1.3 Connections with Neural Processes

Stochastic processes offer a natural formalism to specify priors over functions. Formally, a stochastic process indexed by $x \in \mathcal{X}$ is a collection of random variables $\{F(x)\}_{x \in \mathcal{X}}$ such that, for any finite set of indices $x_1, \dots, x_n$, the random vector $F_{1:n} = (F(x_1), \dots, F(x_n))$ has a well-defined joint distribution $p(F_{1:n})$. These joint distributions are the *finite-dimensional marginals* of the process, and by the Kolmogorov extension theorem a family of such marginals that is consistent and exchangeable defines a unique process over the whole index set $\mathcal{X}$. The Gaussian process of Section 31.3 is the canonical example, its finite-dimensional marginals being Gaussians fully determined by a mean and a covariance function. The question is under what conditions the construction of a function-space prior in Equation 31.3 yields a stochastic process. The literature on stochastic processes provides a complete answer to this, based on the necessity to satisfy exchangeability and consistency [862]. Exchangeability requires invariance of the joint distribution with respect to permutations of the inputs:

$$p(F_{1:n}) = p(\tilde{F}_{1:n}) \quad (31.4)$$

where we denoted by $\tilde{F}_{1:n}$ the permuted versions of $F_{1:n}$. Consistency requires some form of “closure” with respect to marginalisation; in particular, we require that:

$$p(F_{1:m}) = \int p(F_{1:n}) dF_{m+1:n} \quad (31.5)$$

For a fixed prior over θ and a measurable deterministic neural network, the finite-dimensional marginals induced by Equation 31.3 are permutation invariant and marginally consistent, and therefore define a stochastic process in this formal sense. This observation is conceptually useful, but it does not remove the main computational issue: the finite-dimensional densities are still generally unavailable in closed form. In the literature, a construction making use of NNs to learn process-like finite-dimensional distributions has been proposed in Garnelo et al. [582]. These so-called Neural Processes possess some interesting properties, as they allow for their direct use as function-space priors. Intuitively, a neural process learns to behave like a stochastic process directly: given a *context* set of input–output pairs, an encoder network summarises them into a global latent variable, and a decoder network maps this latent variable, together with a target input, to a predictive distribution over the corresponding output. Sharing the latent variable across all target inputs is what couples the predictions and lends the model its process-like, internally consistent behaviour, while training over many randomly sampled context/target splits encourages (approximate) exchangeability and consistency. The result is a flexible, learnable prior over functions whose finite-dimensional marginals are produced by a single forward pass, rather than by marginalising a prior over weights

as in Equation 31.3. In the remainder of this chapter, however, we do not need to use the full stochastic-process structure explicitly in order to determine priors over parameters.

31.1.4 Sampling from function-space priors

The intractability of the function-space prior Equation 31.3 does not prevent us from being able to obtain samples from it. Viewing Bayesian NNs as hierarchical models characterised by $p(\theta)$ and $p(F_{1:n} | \theta)$, it is a simple matter to set up a sampling scheme to obtain samples of $f(x, \theta)$ for a finite set of locations $x_1, \dots, x_n$. This can be achieved by iteratively sampling $\theta' \sim p(\theta)$ and then $F_i \sim p(F_i | \theta')$, that is simply $f(x_i, \theta')$ for $i = 1, \dots, n$. By repeating the sampling of θ' and by evaluating the corresponding $f(x_i, \theta')$ for $i = 1, \dots, n$, we obtain samples from the joint distribution $(\theta, F_{1:n})$; we can then simply disregard samples from θ to be left with samples from the marginal distribution $p(F_{1:n})$.

31.2 Pathologies of function-space priors

Having established that the function-space prior density $p(F_{1:n})$ of Equation 31.3 is generally analytically intractable, so that we can sample from it but not evaluate it in closed form, we are now going to study the behaviour of function-space priors. This analysis aims at highlighting certain pathologies which emerge from making simple and seemingly benign choices for the priors over the parameters θ . In particular, we study function-space priors for various NN architectures, by varying their depth.

31.2.1 Possible priors over θ

The most common choice is an independent Gaussian prior over the weights, $\theta \sim \mathcal{N}(0, \sigma^2 I)$. More flexible alternatives include *hierarchical* Gaussian priors, which place hyper-priors on the per-layer variances and thereby induce heavier-tailed weight marginals, and *normalising-flow* priors [1628], which transform a simple base distribution through an invertible neural network to obtain a far richer family of weight distributions. We return to these last two families in Section 31.3, where their flexibility is exploited to match a target function-space prior; in the present section, which is concerned only with illustrating pathologies, we focus on the simplest Gaussian case. In all cases, we normalise by scaling the prior variance with the dimension of the preceding layer. This is inspired by the literature on initialisation of NN parameters to avoid saturation effects which induce exploding/vanishing gradients (see, e.g., He et al. [720]). The top row of Figure 31.2 illustrates the functions generated by a fully-connected network with tanh activations whose weights are drawn from a fan-in-scaled Gaussian prior with unit base variance, for increasing depth (2, 4, and 8 layers). Increasing depth results in progressively more pathological functions. Interestingly, the marginal variance of the function-space prior can be kept under control, but the covariance of the realisation of the function $f(x, \theta)$ at different inputs becomes increasingly large, leading to nearly-constant functions. This is hardly desirable in any applications where we expect to observe some interesting behaviour in the data. This prior is putting a considerable amount of emphasis on “uninteresting” nearly-constant functions; we will discuss how to tackle this problem in Section 31.3.

Increasing the width of each layer results in function-space priors resembling Gaussian processes. This can be verified by looking at finite-dimensional distributions of $f(x, \theta)$ at any fixed collection of inputs, which become increasingly Gaussian as the width increases [386, 1227]. However, this effect can only be seen in shallow architectures unless skip connections are introduced [481].

31.3 Matching neural network priors with target function-space priors

In this section, we study ways in which we can impose priors over θ so that the induced distribution over the output of the network matches some target function-space prior of interest. The only working assumption we are going to make is that we have easy access to samples from the target function-space prior. This makes the approach presented here quite general and applicable in a wide range of scenarios.

31.3.1 Candidates for function-space priors

In this section we propose a few ideas of possible function-space priors for which it is easy to obtain samples, without necessarily having access to their analytical form.

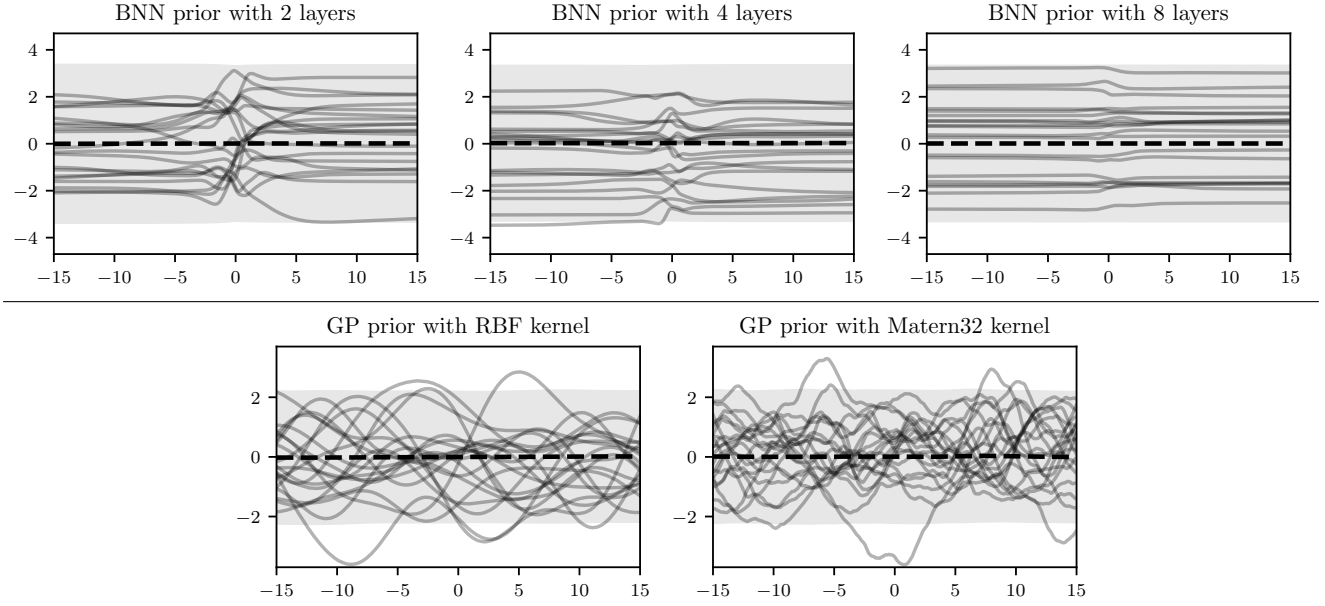


Figure 31.2: (Top) Sample functions of a fully-connected BNN with (left to right) 2, 4, and 8 layers, obtained by placing a fan-in-scaled Gaussian prior with unit base variance on the weights with tanh activations; increasing depth drives the prior towards progressively more pathological, nearly-constant functions. (Bottom) Samples from a GP prior with an RBF (left) and a Matérn-3/2 (right) kernel, which retain a controlled length-scale and marginal variance. The horizontal axis is the one-dimensional input and the vertical axis the function value. The two rows are shown together to contrast the pathologies of deep BNN priors with the well-behaved GP priors.

Gaussian process priors

As a candidate for an easy-to-interpret function-space prior we can use Gaussian processes, for which sampling is easy. In particular, considering again a set of n input points $x_i \in \mathcal{X} \subset \mathbb{R}^D$, we can define a Gaussian process through the choice of a covariance function $k(\cdot, \cdot)$ and a mean function $\mu(\cdot)$. These functions define the behaviour and properties of samples in the function space, as illustrated in the second row of Figure 31.2. In order to draw samples from the Gaussian process prior, we first calculate the covariance matrix K with entries $k_{ij} = k(x_i, x_j)$ and the mean vector with entries $\mu_i = \mu(x_i)$, and then we factorise it using the Cholesky decomposition as $K = LL^\top$. The lower triangular factor L allows us to obtain one sample from the Gaussian process prior by drawing n independent Gaussian random variables $\nu \sim \mathcal{N}(0, I)$ and then by setting $f = L\nu + \mu$. We can repeat this process as many times as needed to obtain multiple samples f from the Gaussian process prior. Once K is factorised (cost in $\mathcal{O}(n^3)$), the cost of drawing one sample is in $\mathcal{O}(n^2)$. Gaussian processes also have the added benefit of providing an analytical form for their density, but we will not use this specifically here.

Physics-based simulations

Beyond explicit function-space priors like Gaussian processes, physics-based simulations offer a sophisticated means to define function-space priors, especially when the underlying system dynamics are governed by known physical laws. These simulations can be driven by differential equations (see, e.g., Lorenzi and Filippone [1079]) and specific boundary/initial conditions, intrinsically constrain the space of admissible functions to those that are physically consistent and plausible. By running these simulations under varied, yet physically realistic, input parameters or initial states, one can generate prior samples in function space that implicitly encode the structure, symmetries, and conservation laws dictated by the physics [774, 1570]. This approach is particularly valuable when analytical forms of function-space priors are unavailable or overly simplistic, allowing for the sampling of complex functions that capture the true behaviour of the system, thus providing a rich and informative prior for neural networks [1371].

The case of unsupervised learning

To highlight the generality of this approach, we note that choosing a prior over θ from target distribution *samples* extends naturally from supervised to unsupervised learning. If we consider unsupervised learning tasks, NNs can

be used to model the true underlying probability distribution over the inputs $\pi(x)$ or to obtain a model to sample from $\pi(x)$ without explicitly modelling it. In these cases, NNs implement functions $f(x, \theta)$ aiming at reconstructing the inputs themselves. The function-space prior of interest is then $\pi(x)$ and the samples are just the given input data x_i , as studied in Tran et al. [1627] for Bayesian Auto-Encoders. The point for our purposes is that, exactly as in the supervised case, the matching objectives of Section 31.3 require only samples from the target, here the data themselves, and never an analytical form for it.

31.3.2 Matching problem and objectives

Assuming that we have access to samples from a function-space prior of interest, say $p_{\text{target}}(F_{1:n})$, we are interested in choosing a prior over θ so that the induced prior $p_{\text{nn}}(F_{1:n})$ (where the subscript “nn” stands for “neural network”) matches the target function-space prior $p_{\text{target}}(F_{1:n})$ as closely as possible. If we only have access to samples from $p_{\text{target}}(F_{1:n})$, we can only hope to establish a close match conditioned on a set of input locations $x_1, \dots, x_n$; expecting a perfect match of the two distributions when n is large is generally unrealistic.

Aiming for a “soft” match of these distributions, we now study objective functions measuring the dissimilarity between distributions over functions evaluated at n input locations. Let’s denote by ψ the parameters (also known as hyper-parameters) of the prior over θ , that is $p(\theta | \psi)$. For example, for a Gaussian prior, ψ could include its mean and variance. The induced function-space prior in Equation 31.3 now reads:

$$p_{\text{nn}}(F_{1:n} | \psi) = \int p(F_{1:n} | \theta) p(\theta | \psi) d\theta. \quad (31.6)$$

Our goal is to minimise an objective measuring the dissimilarity between $p_{\text{nn}}(F_{1:n} | \psi)$ and the target function-space prior $p_{\text{target}}(F_{1:n})$ as follows:

$$\arg \min_{\psi} \{ \mathcal{D}[p_{\text{nn}}(F_{1:n} | \psi), p_{\text{target}}(F_{1:n})] \}, \quad (31.7)$$

where $\mathcal{D}(\cdot, \cdot)$ represents a dissimilarity measure between the two arguments.

We begin this section by choosing $\mathcal{D}(\cdot, \cdot)$ to be the Kullback-Leibler divergence. As we will see shortly, the difficulties stemming from the use of this divergence motivate the need to consider alternatives, and we will present two integral probability metrics [1207] for this purpose, namely the Wasserstein distance and the Maximum Mean Discrepancy.

KL Divergence

The KL divergence objective reads:

$$\mathcal{D}_{\text{KL}}[p_{\text{nn}} \| p_{\text{target}}] = \int p_{\text{nn}}(F_{1:n} | \psi) \log \left(\frac{p_{\text{nn}}(F_{1:n} | \psi)}{p_{\text{target}}(F_{1:n})} \right) dF_{1:n} \quad (31.8)$$

The calculation of this KL divergence is analytically intractable and approximating it through samples is notoriously challenging for large-dimensional distributions. In order to see this, we can rewrite the KL divergence as:

$$\mathcal{D}_{\text{KL}}[p_{\text{nn}} \| p_{\text{target}}] = \mathcal{H}[p_{\text{nn}}] - \mathcal{CH}[p_{\text{nn}} \| p_{\text{target}}] \quad (31.9)$$

where $\mathcal{H}$ and $\mathcal{CH}$ denote the entropy and cross-entropy, respectively. Estimators for the entropy of large dimensional distributions through samples generally suffer from high variance [403], making it difficult to optimise the KL objective with respect to ψ . Similar issues arise in the case of f -divergences and α -divergences. An alternative is to consider other measures of dissimilarity which are easy to estimate based on samples from the two distributions. This is what we discuss next by considering two integral probability metrics.

Wasserstein Distance

The Wasserstein distance is a popular integral probability metric frequently used in optimal transport problems [1679]. Let $\mathcal{F} \subset \mathbb{R}^C$ and $\mathcal{F}' \subset \mathbb{R}^C$ and let $p_1(f)$ and $p_2(f')$ be two (Borel) probability measures. We use the generic symbols p_1 and p_2 while introducing the distance; they will be instantiated as the BNN and target function-space priors, $p_1 = p_{\text{nn}}$ and $p_2 = p_{\text{target}}$, in Equation 31.13 below. The p -Wasserstein distance is defined as¹

$$W_p(p_1, p_2) = \left(\inf_{\gamma \in \Gamma(p_1, p_2)} \int_{\mathcal{F} \times \mathcal{F}'} D(f, f')^p \gamma(f, f') df df' \right)^{1/p}, \quad (31.10)$$

¹The definition of the p -Wasserstein distance holds more generally for any complete separable metric spaces $\mathcal{F}$ and $\mathcal{F}'$.

where $D(f, f')$ is a proper distance metric between $f, f' \in \mathcal{F} \times \mathcal{F}'$ and $\Gamma(p_1, p_2)$ is the set of all joint distributions (couplings) with marginals p_1 and p_2 .

When $\mathcal{F} = \mathcal{F}'$ the most common formulation is the 1-Wasserstein distance with $D(\cdot, \cdot)$ taken to be the Euclidean norm:

$$W(p_1, p_2) = \inf_{\gamma \in \Gamma(p_1, p_2)} \int_{\mathcal{F} \times \mathcal{F}} \|f - f'\| \gamma(f, f') df df', \quad (31.11)$$

The intuitive meaning of this distance is to measure the minimum amount of mass which should be transported in order to transform the distribution p_1 into the distribution p_2 . This is why the 1-Wasserstein distance is also referred to as the Earth-Mover distance.

A convenient rewriting of the 1-Wasserstein distance leads to a dual form which lends itself to a tractable estimation [868, 869]:

$$W(p_1, p_2) = \sup_{\|g\|_L \leq 1} \int_{\mathcal{F}} g(f) p_1(f) df - \int_{\mathcal{F}} g(f') p_2(f') df' \quad (31.12)$$

where g is a 1-Lipschitz continuous function defined on $\mathcal{F} \rightarrow \mathbb{R}$. Indeed, the calculation of the 1-Wasserstein distance can now be viewed as an optimisation problem of a simple objective with respect to 1-Lipschitz continuous functions. The objective is just a difference of two expectations, which can be estimated through Monte Carlo, whereas the optimisation with respect to g can be tackled by parameterising g as a neural network.

The 1-Wasserstein distance can be used as a matching objective between $p_{\text{nn}}(F_{1:n} | \psi)$ and $p_{\text{target}}(F_{1:n})$:

$$W(p_{\text{nn}}, p_{\text{target}}) = \sup_{\|g\|_L \leq 1} \int_{\mathcal{F}} g(f) p_{\text{nn}}(f | \psi) df - \int_{\mathcal{F}} g(f') p_{\text{target}}(f') df' \quad (31.13)$$

Maximum Mean Discrepancy

Another classic example of integral probability metric is the Maximum Mean Discrepancy (MMD) [666]. The presentation of the Wasserstein distance in the previous section allows us to view MMD under a similar vein, as it results in the same objective except that the supremum is taken with respect to functions belonging to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ [48] and bounded norm. The MMD between p_{target} and p_{nn} is defined as follows

$$\text{MMD}(p_{\text{nn}}, p_{\text{target}}) = \sup_{\|g\|_{\mathcal{H}} \leq 1} \int_{\mathcal{F}} g(f) p_{\text{nn}}(f | \psi) df - \int_{\mathcal{F}} g(f') p_{\text{target}}(f') df' \quad (31.14)$$

The Hilbert space $\mathcal{H}$ is induced by the choice of a kernel function $k(\cdot, \cdot)$, and it is desirable to consider kernels inducing infinite dimensional Hilbert spaces, which are capable of capturing discrepancies between an infinite number of moments for the two distributions. By leveraging the so-called kernel trick [48], it is possible to express the square MMD in such a way that does not require $g(f)$ directly, but just scalar products between elements of the RKHS, that is the kernel function evaluated at pairs of f , leading to:

$$\text{MMD}^2(p_{\text{nn}}, p_{\text{target}}) = \mathbb{E}_{f, f' \sim p_{\text{target}}, h, h' \sim p_{\text{nn}}(\cdot | \psi)} [k(f, f') + k(h, h') - 2k(f, h')]. \quad (31.15)$$

All these expectations can be estimated in an unbiased fashion through Monte Carlo by using samples from p_{target} and p_{nn} . Furthermore, this formulation lends itself to mini-batching, while preserving unbiasedness.

31.3.3 Imposing GP Priors on Bayesian Neural Networks using Wasserstein Distance

This section presents a practical implementation for imposing GP priors on Bayesian neural networks in function space, using the previously introduced 1-Wasserstein distance. We match the two stochastic processes at measurement points $Z = [z_1, \dots, z_M]^\top$ sampled from a distribution $q(z)$. For low-dimensional inputs, Z can be drawn from a grid or uniformly sampled. In higher dimensions, samples can be taken from the training set, optionally augmented with noise. If the test input region is known, $q(z)$ can be chosen to cover it.

To match the BNN function-space prior p_{nn} to the target GP prior p_{gp} at these measurement points, we employ a sample-based approach that uses the 1-Wasserstein distance of Equation 31.13 as the objective:

$$\min_{\psi} \max_{\omega} \mathbb{E}_q \left[\underbrace{\mathbb{E}_{p_{\text{nn}}(\cdot | \psi)}[g(f_M, \omega)] - \mathbb{E}_{p_{\text{gp}}}[g(f_M, \omega)]}_{\mathcal{L}(\psi, \omega)} \right], \quad (31.16)$$

where f_M denotes the set of random variables associated with the inputs Z , and $g(\cdot, \omega)$ is a 1-Lipschitz function parameterised by a neural network with parameters ω . The dependence on the prior hyper-parameters ψ enters through the BNN function-space prior $p_{\text{nn}}(\cdot | \psi)$: its samples f_M are produced by first drawing $\theta \sim p(\theta | \psi)$ and then evaluating the network at Z , so that minimising over ψ reshapes the BNN prior to match the target.

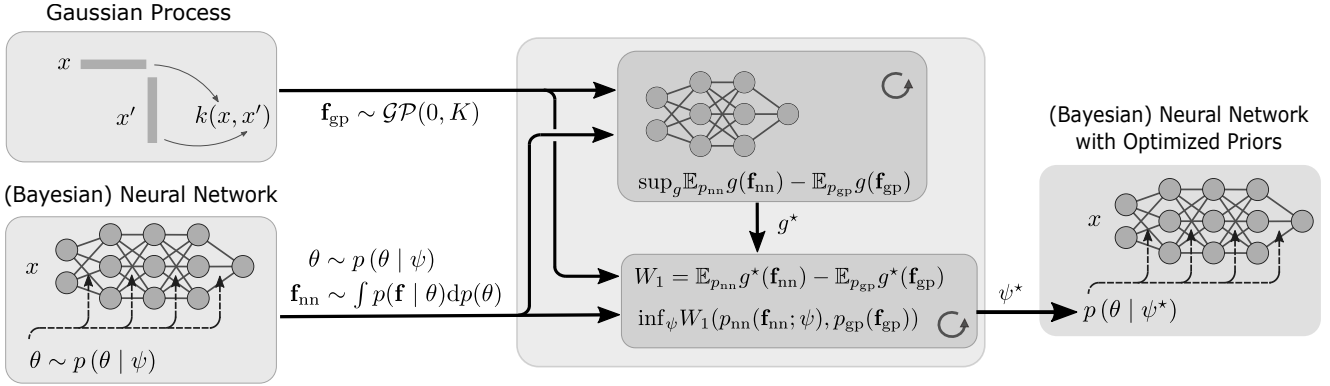


Figure 31.3: Schematic representation of the process of imposing GP priors on BNNs via Wasserstein distance minimisation.

Regarding the optimisation of the ψ and ω parameters, we alternate between $n_{\text{Lipschitz}}$ steps of maximising $\mathcal{L}$ with respect to the Lipschitz function’s parameters ω and one step of minimising the Wasserstein distance with respect to the prior’s parameters ψ . We therefore use two independent optimisers for ψ and ω . Figure 31.3 offers a high-level schematic representation of the proposed procedure. Given samples from two stochastic processes, the Wasserstein distance is estimated by considering the inner maximisation of Equation 31.16, resulting in an optimal g^* . This inner optimisation step is repeated for every step of the outer optimisation loop. Notice that the objective is fully sample-based. As a result, it is not necessary to know the closed-form of the marginal density $p_{\text{nn}}(\mathbf{f}; \psi)$. One may consider any stochastic process as a target prior over functions, as long as we can draw samples from it.

To enforce the Lipschitz constraint on g_ω , Arjovsky et al. [47] propose clipping the weights ω to lie within a compact set $[-c, c]$, ensuring all functions g_ω are K -Lipschitz. However, this often biases g_ω toward overly simple functions. Building on the fact that a differentiable function is 1-Lipschitz if and only if the norm of its gradient is at most one everywhere, Gulrajani et al. [686] propose penalising deviations of the gradient norm from 1. Specifically, they augment the loss with a gradient penalty term:

$$\mathcal{L}_R(\psi, \omega) = \mathcal{L}(\psi, \omega) + \lambda \mathbb{E}_{p_{\hat{\mathbf{f}}}} \left[\left(\left\| \nabla_{\hat{\mathbf{f}}} g(\hat{\mathbf{f}}) \right\|_2 - 1 \right)^2 \right], \quad (31.17)$$

where $p_{\hat{\mathbf{f}}}$ is the distribution of interpolated functions $\hat{\mathbf{f}} = \varepsilon \mathbf{f}_{\text{nn}} + (1 - \varepsilon) \mathbf{f}_{\text{gp}}$, with $\varepsilon \sim \mathcal{U}[0, 1]$, and $\mathbf{f}_{\text{nn}} \sim p_{\text{nn}}$, $\mathbf{f}_{\text{gp}} \sim p_{\text{gp}}$ drawn from BNN and GP priors, respectively. The coefficient λ controls the strength of the regularisation.

31.3.4 Examples on a 1D Regression Synthetic Dataset

The dataset is constructed as follows: (1) uniformly sample 64 input locations $\mathbf{x}$ in the interval $[-10, 10]$; (2) rearrange these locations to introduce a gap in the input space; (3) sample function values $\mathbf{f}$ from a GP prior with lengthscale $l = 0.6$ and amplitude $\alpha = 1$, evaluated at $\mathbf{x}$; (4) corrupt the targets with i.i.d. Gaussian noise with variance $\sigma_\epsilon^2 = 0.1$. We use a three-layer MLP as the model. Figure 31.4 presents the results, organised in five rows, with the reference Gaussian process shown in the leftmost column throughout. We consider three GP-induced (GPi) BNN priors: GPi-Gaussian, GPi-Hierarchical, and GPi-NFlow, whose prior over θ is parameterised by Gaussian, hierarchical, and normalising-flow [1628] distributions, respectively. The first row shows samples from these priors with their hyper-parameters kept *fixed* at sensible default values, that is *not* matched to the GP (labelled “Fixed”), whereas the second row shows samples after the hyper-parameters ψ have been *optimised* to match the GP prior by minimising the Wasserstein distance (labelled “Optim”). The third row reports the convergence of this prior-matching optimisation. The last two rows show the posteriors obtained with the fixed and the optimised priors, respectively.

The procedure is therefore two-stage, and the two stages should not be conflated. In the first (prior-matching) stage we solve the optimisation problem of Equation 31.16 to obtain the prior hyper-parameters ψ^* ; this stage

operates purely in function space and only requires *samples* from p_{nn} and p_{gp} . In the second (inference) stage we fix $\psi = \psi^*$ and draw posterior samples with stochastic gradient Hamiltonian Monte Carlo (SGHMC; Chapter 2) [273]. It is worth emphasising why the second stage is unproblematic even though the *function-space* prior density is intractable: posterior sampling is carried out in *parameter* space, where the prior $p(\theta \mid \psi^*)$, a Gaussian, hierarchical-Gaussian, or normalising-flow density over θ , is available in closed form and is differentiable. The sampler thus targets the unnormalised parameter-space posterior $p(y \mid x, \theta) p(\theta \mid \psi^*)$ and never needs to evaluate the intractable marginal $p_{\text{nn}}(F_{1:n})$; the optimisation of Equation 31.16 is performed once, beforehand, and is not part of the sampler's inner loop.

From the analysis of these plots, we clearly see the benefit of placing a prior on the functions rather than on the parameters. First, the Wasserstein distance plots show satisfactory convergence, with the normalising-flow prior closely matching the GP prior thanks to its flexibility. Second, as expected, the posteriors exhibit similar behaviour according to the possible solutions realisable from the prior: classic priors tend to yield degenerate functions resulting in overconfidence in regions without data, while our GP-induced priors retain information regarding lengthscale and amplitude.

Overall, the examples show why parameter-space convenience is not enough for Bayesian deep learning: apparently innocuous weight priors can induce highly restrictive priors over functions. Matching in function space provides a practical way to encode interpretable assumptions such as length-scale, amplitude, and smoothness while still carrying out posterior inference in parameter space.

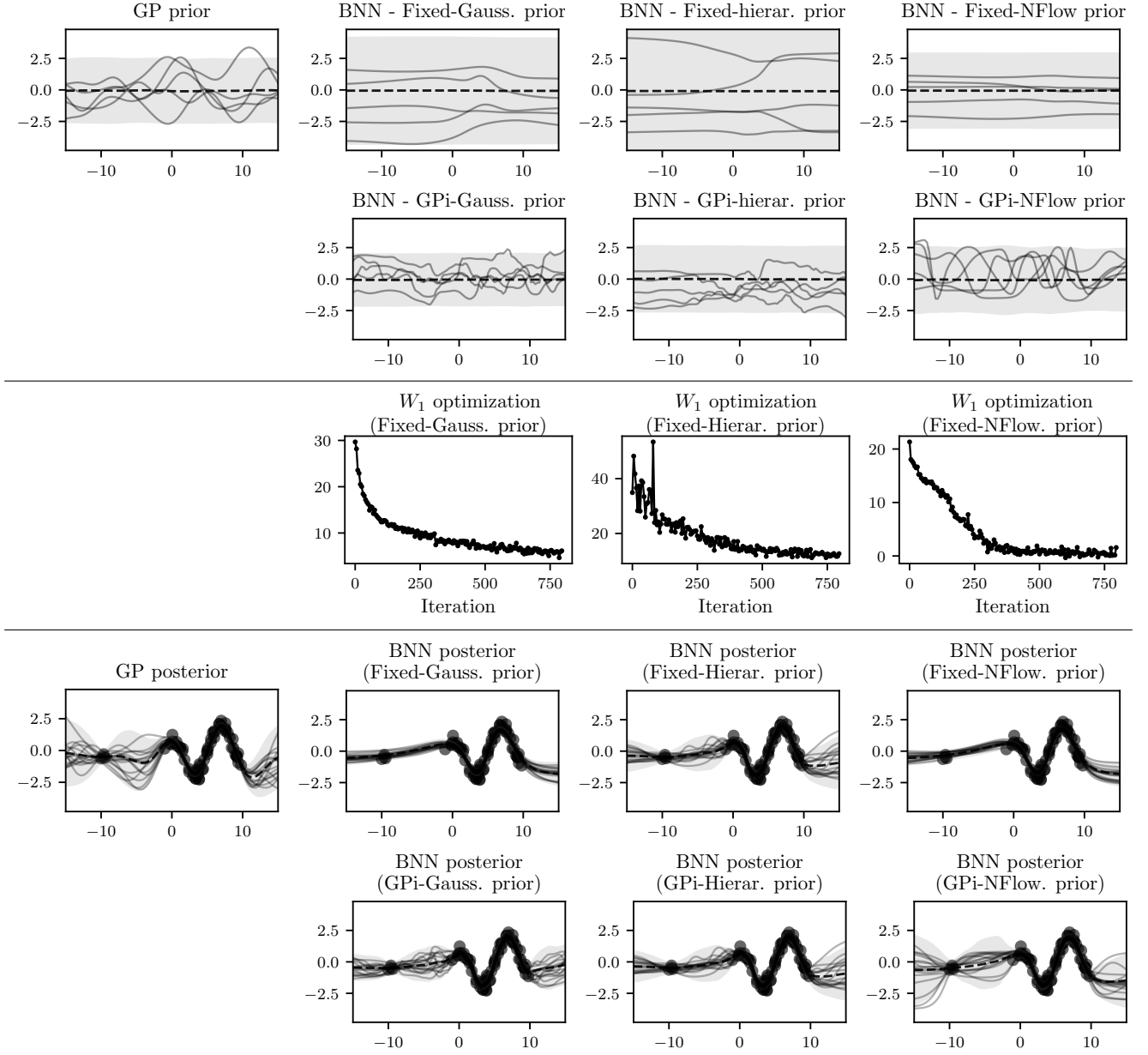


Figure 31.4: Visualisation of a one-dimensional regression example with a three-layer MLP. Throughout, the left column is the reference Gaussian process and the horizontal/vertical axes are the input and the function value, respectively. The first row shows BNN prior samples with *fixed* (default, unmatched) hyper-parameters and the second row with hyper-parameters *optimised* to match the GP prior; the middle row shows the convergence of this prior-matching optimisation. The last two rows show the posteriors obtained, respectively, with the fixed and the optimised priors, sampled with SGHMC. The dashed line and shaded areas denote the posterior mean and the 95% credible intervals.

Part VII

Identifiability and symmetries

Chapter 32

Symmetries in Bayesian neural networks

Symmetries are transformations that leave a system unchanged. These transformations may preserve the whole system or only certain properties of it. In many areas of science, especially mathematics and physics, symmetries are essential because they allow us to simplify complex problems and understand systems more easily.

One of the most famous results on symmetries in physics is **Noether’s theorem** [1247]: every continuous symmetry of a physical system corresponds to a conserved quantity, as when time-invariance of the laws of physics yields conservation of energy. Symmetries also make hard equations tractable; for example, by assuming spherical symmetry around a massive object, Schwarzschild obtained the first exact solution to Einstein’s field equations and, with it, the prediction of black holes [1478]. We invoke these only as motivation: the structure that matters in this chapter is the symmetry of the parametrisation, to which we turn now.

In recent years, machine learning—and especially deep learning—has also been shown to involve important symmetries. These appear in two main ways:

1. **Symmetries in the data space:** These are transformations of the input data (and possibly the labels) that should not change the model’s output. For example, in image classification, moving a digit within an image (like in the MNIST dataset) should not change the predicted label. This kind of symmetry is the focus of *geometric deep learning* [210].
2. **Symmetries in the parameter space:** These occur in the way neural networks are built. Deep models use layers of weights (matrices) and activation functions like ReLU or tanh. In many cases, different arrangements or values of weights can produce exactly the same output function. These transformations in parameter space do not affect the final result of the model but can cause redundancy during training or evaluation [493].

In the context of *Bayesian deep learning* [174, 1105], handling symmetries is crucial. It is important to be precise about what is at stake: in *exact* Bayesian model averaging the predictive distribution lives in function space, and it is not harmed by a function having several parameter representatives, since these representatives all contribute the same predictions. The practical difficulties are instead inherited by the tools we use to *approximate* the posterior. Ignoring symmetries can make weight-space posterior approximations and convergence diagnostics misleading, and can make sampling or variational inference inefficient, because the same solution is revisited many times in different but equivalent forms. Accounting for symmetries therefore makes weight-space inference more efficient and easier to interpret.

The Bayesian message. A single thread runs through this chapter. A group G of function-preserving transformations acts on the parameter space; its orbits are the equivalence classes of parameters that implement the same function, and along each orbit both the network map and the likelihood are constant. Whether the *posterior* is also constant along an orbit is a separate question: it requires that the prior *measure* (prior density together with the base measure) be invariant under the transformation as well. Permutation symmetries are typically posterior symmetries under exchangeable priors, whereas continuous scaling symmetries usually preserve the likelihood but not a standard Gaussian prior (L2 regularisation). Keeping likelihood symmetries and posterior symmetries distinct organizes the discussion of sampling, variational inference, ensembles, and mode connectivity that follows.

Structure of the Chapter. This chapter is organized as follows: First, we introduce the general concepts of **groups, symmetries, and invariance**, and explain how they relate to machine learning. Next, we focus on **weight-space symmetries** in deep neural networks, showing how different parameter configurations can represent the same function. Then, we explore various approaches to **handle symmetries by choosing a representative** of each equivalence class, especially in Bayesian settings, including symmetry-aware optimisation and sampling. Finally, we discuss **mode connectivity** and its implications for understanding the structure of the loss landscape.

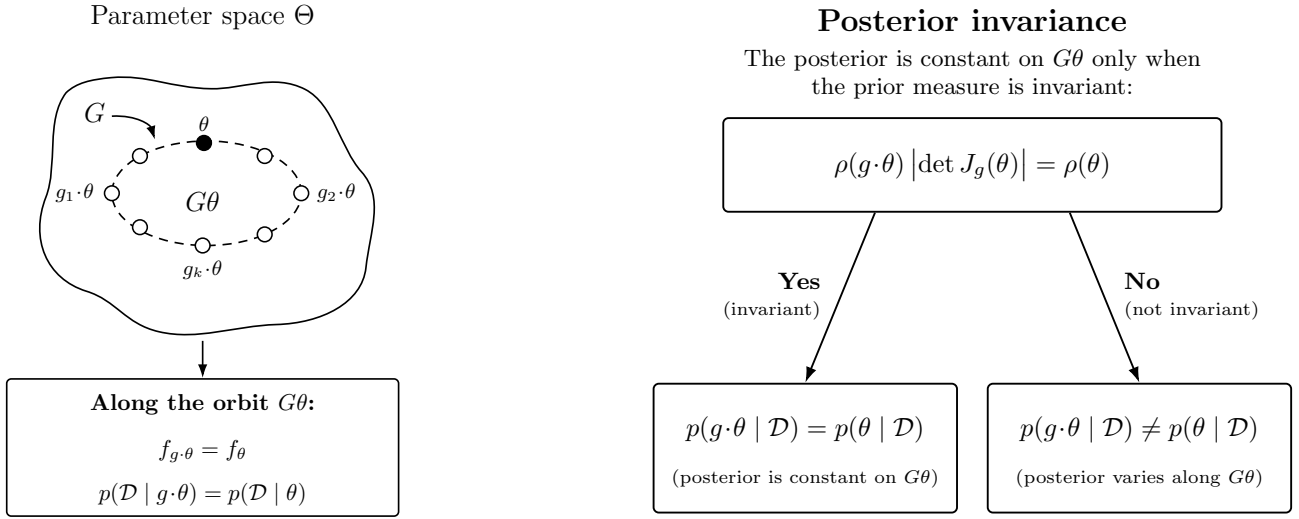


Figure 32.1: Likelihood and posterior invariance along a parameter-space orbit. The network function and likelihood are constant on $G \cdot \theta$, while posterior invariance additionally requires invariance of the prior measure.

We study the consequences of these symmetries for posterior sampling in chapter 33 and their identification in chapter 34.

32.1 Background: symmetries and invariance

32.1.1 Basic definitions

We start by defining the basic concepts that will be used throughout this chapter and beyond. We first recall the definitions of groups and subgroups.

Definition 32.1 (Group). A group is a non-empty set G together with a binary operation $\cdot : G \times G \rightarrow G$ (often written as $a \cdot b$) that satisfies the following axioms for all $a, b, c \in G$:

- **Associativity:** $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity element:** There exists an element $e \in G$ such that $e \cdot a = a \cdot e = a$ for all $a \in G$. This element e is called the identity element.
- **Inverse element:** For every $a \in G$, there exists an element $b \in G$ such that $a \cdot b = b \cdot a = e$, where e is the identity element. The element b is called the inverse of a and is often denoted a^{-1} .

Definition 32.2 (Subgroup). Let G be a group. A subset $H \subseteq G$ is called a subgroup of G if H is itself a group under the group operation inherited from G .

The subgroups G and $\{e\}$, where e is the identity element of G , are called the *improper subgroups* of G . All other subgroups are called *proper subgroups*.

Theorem 32.1. A nonempty subset H of a group G is a subgroup if and only if the following two conditions hold:

1. If $h, k \in H$, then $h \cdot k \in H$.
2. If $h \in H$, then $h^{-1} \in H$.

Proof. If H is a subgroup, then it must satisfy the group axioms, so conditions (1) and (2) clearly hold.

Conversely, suppose H satisfies conditions (1) and (2). We show that H is a group under the operation inherited from G .

Since G is a group, the operation is associative, and thus associativity also holds in H .

Because H is nonempty, there exists some $h \in H$. By condition (2), $h^{-1} \in H$. Then, by condition (1), $h \cdot h^{-1} = e \in H$, so the identity element e belongs to H .

Thus, H contains the identity, is closed under the group operation, and contains inverses of its elements. Therefore, H is a subgroup of G . $\square$

32.1.2 Symmetric groups

More specifically, we now define the symmetric group, which will be needed in the next parts.

Definition 32.3 (Symmetry). *A symmetry of a set X is a bijective transformation $f : X \rightarrow X$ such that f preserves the structure of X .*

The notion of "structure" depends on the context:

- In **geometry**, the structure refers to *shape and size*.
- In **algebra**, it refers to *operations* (e.g., a group operation).
- In **physics**, it refers to *invariance under a transformation* (e.g., time or space translation).

Definition 32.4 (Permutation in $\mathbb{R}^n$). *Let $x = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n$. A permutation of x is a vector $\sigma(x) = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)})$, where σ is a bijection from the set $\{1, 2, \dots, n\}$, denoted as $[n]$, to itself.*

Each permutation in $\mathbb{R}^n$ σ can be represented by an $n \times n$ permutation matrix π_σ , which is obtained by permuting the rows (or columns) of the identity matrix according to σ .

Example: Let σ be a permutation in $\mathbb{R}^3$ defined by

$$\sigma(1) = 2, \quad \sigma(2) = 3, \quad \sigma(3) = 1.$$

Then the corresponding permutation matrix π_σ is:

$$\pi_\sigma = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

The action of σ on a column vector $x \in \mathbb{R}^3$ is given by matrix multiplication:

$$\sigma(x) = \pi_\sigma x.$$

Definition 32.5 (Symmetric Group). *Let $n \in \mathbb{N}$. The symmetric group on n elements, denoted by $\mathfrak{S}_n$, is the set of all bijections (or permutations) from the set $\{1, 2, \dots, n\}$ to itself. That is,*

$$\mathfrak{S}_n = \{\sigma : [n] \rightarrow [n] \mid \sigma \text{ is a bijection}\}.$$

The group operation on $\mathfrak{S}_n$ is the composition of functions: for $\sigma, \tau \in \mathfrak{S}_n$, the product $\sigma \circ \tau$ is the bijection defined by

$$(\sigma \circ \tau)(i) = \sigma(\tau(i)), \quad \text{for all } i \in [n].$$

Proposition 32.1. *The symmetric group $\mathfrak{S}_n$, under composition of functions, forms a group: the identity element is the identity function, and each permutation has an inverse given by its inverse function.*

Proof. To show that $\mathfrak{S}_n$ forms a group, we verify the group axioms:

- **Closure:** Let $\sigma, \tau \in \mathfrak{S}_n$. Then σ and τ are bijections from $[n]$ to itself. The composition $\sigma \circ \tau$ is also a bijection, and hence $\sigma \circ \tau \in \mathfrak{S}_n$.
- **Associativity:** Let $\sigma, \tau, \rho \in \mathfrak{S}_n$ be permutations. We must show that function composition is associative; that is,

$$(\sigma \circ \tau) \circ \rho = \sigma \circ (\tau \circ \rho).$$

Let $i \in \{1, 2, \dots, n\}$ be arbitrary. We compute both sides of the equation evaluated at i :

$$\begin{aligned} ((\sigma \circ \tau) \circ \rho)(i) &= (\sigma \circ \tau)(\rho(i)) \\ &= \sigma(\tau(\rho(i))), \end{aligned}$$

and

$$\begin{aligned} (\sigma \circ (\tau \circ \rho))(i) &= \sigma((\tau \circ \rho)(i)) \\ &= \sigma(\tau(\rho(i))). \end{aligned}$$

Since both expressions evaluate to $\sigma(\tau(\rho(i)))$ for all i , we conclude that

$$(\sigma \circ \tau) \circ \rho = \sigma \circ (\tau \circ \rho),$$

as functions.

Therefore, function composition is associative in $\mathfrak{S}_n$, and the associativity axiom holds.

- **Identity element:** The identity function $\text{id}: [n] \rightarrow [n]$, defined by $\text{id}(i) = i$ for all i , is a bijection and thus belongs to $\mathfrak{S}_n$. For any $\sigma \in \mathfrak{S}_n$,

$$\sigma \circ \text{id} = \text{id} \circ \sigma = \sigma.$$

- **Inverses:** Since $\sigma \in \mathfrak{S}_n$ is a bijection, it has an inverse function $\sigma^{-1}: [n] \rightarrow [n]$ such that

$$\sigma \circ \sigma^{-1} = \sigma^{-1} \circ \sigma = \text{id}.$$

Because σ^{-1} is also a bijection, we have $\sigma^{-1} \in \mathfrak{S}_n$.

Thus, $\mathfrak{S}_n$ satisfies all the group axioms and is therefore a group under composition of functions. $\square$

32.1.3 Group actions, invariant functions, and equivariant functions

In this chapter, we will often refer to equivariant functions, we formalise and introduce them below.

Definition 32.6 (Group Action). *Let G be a group and X a set. A (left) group action of G on X is a function*

$$\cdot: G \times X \rightarrow X, \quad (g, x) \mapsto g \cdot x,$$

satisfying the following two properties for all $g, h \in G$ and all $x \in X$:

1. **Identity:** $e \cdot x = x$, where $e \in G$ is the identity element.
2. **Compatibility:** $(gh) \cdot x = g \cdot (h \cdot x)$.

We then say that G acts on X , and X is called a G -set.

Definition 32.7 (Orbit). *The orbit of an element $x \in X$ under the action of a group G , denoted $G \cdot x$, is the set of all elements in X to which x can be moved by the action of elements in G : $G \cdot x = \{g \cdot x \mid g \in G\}$.*

Definition 32.8 (Stabilizer). *The stabilizer of an element $x \in X$ is the subgroup $G_x = \{g \in G \mid g \cdot x = x\}$ of transformations that leave x fixed.*

Definition 32.9 (Quotient set). *The set of orbits $X/G = \{G \cdot x \mid x \in X\}$ is called the quotient of X by the action of G . Choosing one element from each orbit defines a set of representatives, and a map sending each x to its representative is a canonicalisation.*

Definition 32.10 (G -invariant Function). *Let G be a group acting on a set X , and let $f: X \rightarrow Y$ be a function into some set Y . We say that f is G -invariant (or invariant under the action of G) if for all $g \in G$ and all $x \in X$, we have*

$$f(g \cdot x) = f(x).$$

In other words, f is constant on the orbits of the group action.

Definition 32.11 (G -Equivariant Function). *Let G be a group acting on two sets X and Y via actions*

$$\cdot_X: G \times X \rightarrow X, \quad (g, x) \mapsto g \cdot_X x,$$

and

$$\cdot_Y: G \times Y \rightarrow Y, \quad (g, y) \mapsto g \cdot_Y y.$$

A function $f: X \rightarrow Y$ is said to be G -equivariant if for every $g \in G$ and every $x \in X$,

$$f(g \cdot_X x) = g \cdot_Y f(x).$$

That is, applying a group element before or after f yields the same result.

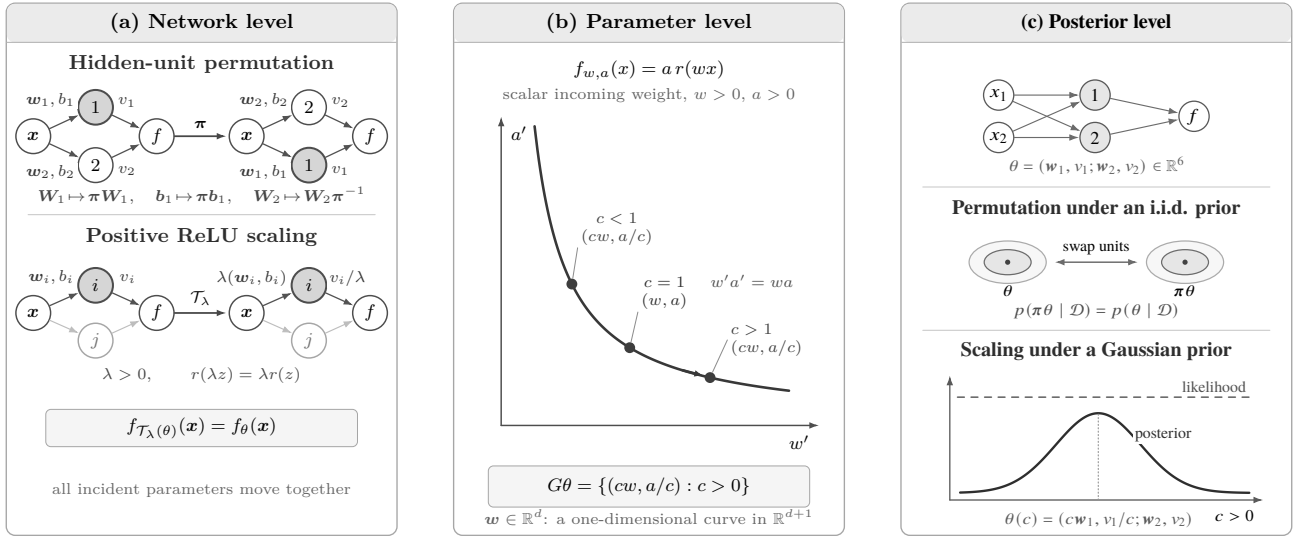


Figure 32.2: Three views of hidden-unit symmetries in a one-hidden-layer ReLU MLP. (a) A permutation reorders each hidden unit together with all its incident parameters. Positive scaling sends (w_i, b_i, v_i) to $(\lambda w_i, \lambda b_i, v_i/\lambda)$ for $\lambda > 0$, leaving the implemented function unchanged. (b) For $f_{w,a}(x) = a r(wx)$ with $w, a > 0$, the equivalent parameters $(cw, a/c)$ form a branch of the hyperbola $w'a' = wa$. For vector-valued incoming weights, this remains a one-dimensional curve in a higher-dimensional parameter space. (c) In a 2–2–1 BNN without biases, $\theta = (w_1, v_1; w_2, v_2) \in \mathbb{R}^6$. An exchangeable prior yields identical posterior copies under hidden-unit permutation. Along the ReLU scaling orbit, however, the likelihood is constant while the posterior density generally varies under an independent Gaussian prior.

32.2 Going deep into weight-space symmetries in neural networks

There are two main mathematical frameworks for working on parameter-space symmetries. Most papers use the simpler but less general matrix formalism, describing symmetries as transformations of matrices, although some authors prefer the directed acyclic graph formalism. This chapter summarizes the existing literature, presenting the results and intuitions in the matrix formalism.

With the group-theoretic vocabulary of the previous section in place, we can state the unifying picture. Once we restrict attention to invertible transformations, the weight-space symmetries of a fixed architecture form a group G acting on the parameter space $\mathbb{R}^d$, and the map $\theta \mapsto f_\theta$ is G -invariant: it is constant on each orbit $G \cdot \theta$. The orbits are exactly the sets of parameters that implement the same function. Removing redundancy then amounts to choosing one representative per orbit, i.e. working on the quotient $\mathbb{R}^d/G$.

Definition 32.12 (Weight-space symmetries). *Fix a network architecture, and let f_θ denote the function it implements with parameter vector $\theta \in \mathbb{R}^d$. A weight-space symmetry of this architecture is an invertible map $\mathcal{T} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ that preserves the implemented function for every parameter vector, that is,*

$$\forall \theta \in \mathbb{R}^d, \forall x \in \mathcal{X} : f_\theta(x) = f_{\mathcal{T}(\theta)}(x). \quad (32.1)$$

Equivalently, $f_{\mathcal{T}(\theta)} = f_\theta$ as functions. The set of all such maps forms a group under composition, the weight-space symmetry group of the architecture.

Notation. Throughout, $P_m \subset \mathbb{R}^{m \times m}$ denotes the set of $m \times m$ permutation matrices, i.e. the matrix representation of the symmetric group $\mathfrak{S}_m$; a permutation matrix π_σ acts on a vector by $\pi_\sigma x = (x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(m)})^\top$ and is orthogonal, so $\pi_\sigma^{-1} = \pi_\sigma^\top$. We use the same notation P_h for $h \times h$ permutation matrices when permuting attention heads. We reserve σ for permutations and write $\hat{\sigma}$ (always hatted) for an estimated standard deviation. We write $\circ$ for the Hadamard (element-wise) product, following the book convention.

32.2.1 Weight-space symmetries in neural networks

The most important weight-space symmetries in deep neural networks arise either as a permutation of hidden units, which reorders the neurons of a layer, or as a rescaling of the outputs of a layer, which we refer to as

scaling in the following. These symmetries and some others – also classically encountered in neural networks – are summarised in table 32.1.

Permuting weights

The foundational work on weight-space symmetries is due to Hecht-Nielsen [724], who formally introduced the notion of *permutation symmetries* in feedforward neural networks. They also established an upper bound on the number of such symmetries based on the number of permissible permutations per layer.

Proposition 32.2 (Permutation symmetries). *Consider a one-hidden-layer network*

$$f_\theta(x) = \mathbf{W}_2 r(\mathbf{W}_1 x + b_1) + b_2,$$

with $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$, $b_1 \in \mathbb{R}^m$, $\mathbf{W}_2 \in \mathbb{R}^{k \times m}$, $b_2 \in \mathbb{R}^k$, and an activation r applied element-wise. Let $\pi \in P_m$ be a permutation matrix acting on the m hidden units. Then the transformation

$$\mathbf{W}_1 \mapsto \pi \mathbf{W}_1, \quad b_1 \mapsto \pi b_1, \quad \mathbf{W}_2 \mapsto \mathbf{W}_2 \pi^{-1}, \quad b_2 \mapsto b_2,$$

is a weight-space symmetry: for all $x \in \mathbb{R}^n$,

$$\mathbf{W}_2 \pi^{-1} r(\pi \mathbf{W}_1 x + \pi b_1) + b_2 = \mathbf{W}_2 r(\mathbf{W}_1 x + b_1) + b_2.$$

The construction holds at every hidden layer of a deeper network, so the permutation symmetries of an L -hidden-layer network of widths $m_1, \dots, m_L$ form the finite group $P_{m_1} \times \dots \times P_{m_L}$ of order $\prod_\ell m_\ell!$.

Proof. Write

$$\mathbf{h} = \mathbf{W}_1 x + b_1,$$

so that $f_\theta(x) = \mathbf{W}_2 r(\mathbf{h}) + b_2$. Under the transformation, the pre-activation becomes

$$\mathbf{h}' = \pi \mathbf{W}_1 x + \pi b_1 = \pi(\mathbf{W}_1 x + b_1) = \pi \mathbf{h}.$$

Since π only permutes the entries of $\mathbf{h}$ and r is applied element-wise, the permutation commutes with the activation:

$$r(\pi \mathbf{h}) = \pi r(\mathbf{h}).$$

Hence the transformed output is

$$\mathbf{W}_2 \pi^{-1} r(\pi \mathbf{h}) + b_2 = \mathbf{W}_2 \pi^{-1} \pi r(\mathbf{h}) + b_2 = \mathbf{W}_2 r(\mathbf{h}) + b_2,$$

using $\pi^{-1} \pi = \mathbf{I}_m$. The output is therefore unchanged, completing the proof. $\square$

Scaling weights

The “layer” aspect of deep neural networks not only entails permutation symmetries but also symmetries related to changes in value. Again, the mechanism is simple: the first layer is modified so that the output changes in scale, sign, etc., and the second layer reverses this change. However, contrary to changing the order of the neurons, changes in value may prove impossible to reverse, depending on the chosen activation function.

One early example of symmetries relative to the intensity of the output are *sign-flip symmetries* introduced by Albertini, Sontag, and Maillot [19]. They appear in the case of odd activation functions. Further generalisations to *asymptotically constant activation functions* (excluding ReLU) were proposed by Kůrková [953], who described a broader class of functional symmetries.

At this time, as one can read in Kůrková [953], non-asymptotically constant functions were not considered relevant for training deep neural networks. However, ReLU, which is unbounded for large positive inputs, later emerged as critical, notably due to its efficiency in large-scale computations. The characterisation of scaling symmetries, therefore, needed to be extended. This was done, for instance, by Petzka [1321]:

Proposition 32.3 (Scaling symmetries). *Consider a one-hidden-layer network*

$$f_\theta(x) = \mathbf{W}_2 r(\mathbf{W}_1 x + b_1) + b_2,$$

with $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$, $b_1 \in \mathbb{R}^m$, $\mathbf{W}_2 \in \mathbb{R}^{k \times m}$, $b_2 \in \mathbb{R}^k$, and assume the activation r is positively homogeneous of degree one, i.e., $r(\lambda t) = \lambda r(t)$ for all $\lambda > 0$ and $t \in \mathbb{R}$ (this holds for ReLU and leaky ReLU).

Let $\lambda \in (\mathbb{R}_{>0})^m$ be a vector of positive scaling coefficients and $\mathbf{D} = \text{diag}(\lambda)$. Then the transformation

$$\mathbf{W}_1 \mapsto \mathbf{D}\mathbf{W}_1, \quad b_1 \mapsto \mathbf{D}b_1, \quad \mathbf{W}_2 \mapsto \mathbf{W}_2\mathbf{D}^{-1}, \quad b_2 \mapsto b_2,$$

is a weight-space symmetry: for all $x \in \mathbb{R}^n$,

$$\mathbf{W}_2\mathbf{D}^{-1}r(\mathbf{D}\mathbf{W}_1x + \mathbf{D}b_1) + b_2 = \mathbf{W}_2r(\mathbf{W}_1x + b_1) + b_2.$$

Remark. Degree-one positive homogeneity is essential. A degree-two positively homogeneous activation such as $t \mapsto (t_+)^2$ does *not* yield this symmetry, and negative rescalings ($\lambda < 0$) are not ReLU symmetries because $r(\lambda t) \neq \lambda r(t)$ in general.

Proof. Write

$$\mathbf{h} = \mathbf{W}_1x + b_1,$$

so that $f_\theta(x) = \mathbf{W}_2r(\mathbf{h}) + b_2$. Under the transformation, the pre-activation becomes

$$\mathbf{h}' = \mathbf{D}\mathbf{W}_1x + \mathbf{D}b_1 = \mathbf{D}(\mathbf{W}_1x + b_1) = \mathbf{D}\mathbf{h}.$$

Because $\mathbf{D} = \text{diag}(\lambda)$ with $\lambda > 0$ and r is positively homogeneous of degree one applied element-wise,

$$r(\mathbf{D}\mathbf{h}) = \mathbf{D}r(\mathbf{h}).$$

Therefore the transformed output is

$$\mathbf{W}_2\mathbf{D}^{-1}r(\mathbf{D}\mathbf{h}) + b_2 = \mathbf{W}_2\mathbf{D}^{-1}\mathbf{D}r(\mathbf{h}) + b_2 = \mathbf{W}_2r(\mathbf{h}) + b_2 = f_\theta(x),$$

since $\mathbf{D}^{-1}\mathbf{D} = \mathbf{I}_m$. Thus the network output is unchanged. $\square$

Petzka, Trimmel, and Sminchisescu [1321] give a complete characterisation for fully connected, scalar-output ReLU networks with one hidden layer. After excluding reducible and degenerate representations (including zero units, positively proportional hidden units, coincident activation hyperplanes, and certain exact cancellations), any two parameterisations realizing the same function differ only by permutations of hidden units and positive neuron-wise rescalings. This characterisation does not directly extend to deeper networks [591]: because later ReLU layers operate on the nonnegative image of preceding layers, even nondegenerate parameter configurations can admit additional, parameter-dependent symmetries arising from interactions between layers.

Gegenfurtner, Grillo, and Montúfar [591] precise these effects for generic three-layer ReLU networks in the bottleneck regime. They give a description of the corresponding parameter fibers and identify new continuous and discrete symmetries. In particular, a first-layer activation hyperplane hidden by the second layer may be translated while compensating through the second-layer biases. Relatedly, but without attempting a symmetry classification, Kobińska et al. [908] study how overparameterisation and the prior jointly reshape Bayesian neural-network posteriors. They identify balancedness, weight reallocation along equal-probability manifolds, and prior conformity as three consequences of redundancy, showing that overparameterisation can produce structured, prior-aligned weight-space posteriors.

32.2.2 Weight-space symmetries on special layers

We extend the previous results to different layers to demonstrate that symmetries can also appear independently from the “layer” aspect of deep neural networks, due to the use of specific functions.

Softmax layer

Proposition 32.4 (Softmax symmetries). *In models ending with a softmax layer, additive shifts to the input logits by a constant vector (shared across all samples) do not affect the output probabilities:*

$$\text{softmax}(\mathbf{z} + c\mathbf{1}) = \text{softmax}(\mathbf{z}), \quad \forall c \in \mathbb{R}, \mathbf{z} \in \mathbb{R}^n.$$

Proof. The softmax function is defined for $\mathbf{z} \in \mathbb{R}^n$ by

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}, \quad \text{for } i = 1, \dots, n.$$

Now consider a constant shift $c \in \mathbb{R}$, and let $\mathbf{z}' = \mathbf{z} + c\mathbf{1}$, so that $z'_i = z_i + c$ for all i . Then:

$$\text{softmax}(\mathbf{z}')_i = \frac{e^{z_i+c}}{\sum_{j=1}^n e^{z_j+c}} = \frac{e^c \cdot e^{z_i}}{e^c \cdot \sum_{j=1}^n e^{z_j}} = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} = \text{softmax}(\mathbf{z})_i.$$

Since this holds for each component i , we conclude that: $\text{softmax}(\mathbf{z} + c\mathbf{1}) = \text{softmax}(\mathbf{z})$. $\square$

This logit-shift invariance translates into an unidentifiable direction in the final linear layer. Suppose the pre-softmax logits are produced by $\mathbf{z}(x) = \mathbf{W}\mathbf{h}(x) + \mathbf{b}$ with K classes, where $\mathbf{h}(x)$ is the penultimate feature vector. For any feature-space vector $\mathbf{u}$ and scalar c , the transformation

$$\mathbf{W} \mapsto \mathbf{W} + \mathbf{1}_K \mathbf{u}^\top, \quad \mathbf{b} \mapsto \mathbf{b} + c \mathbf{1}_K$$

adds $(\mathbf{u}^\top \mathbf{h}(x) + c) \mathbf{1}_K$ to every logit vector and therefore leaves the softmax output unchanged. The softmax thus has a $(\dim \mathbf{h} + 1)$ -dimensional family of equivalent linear heads. Standard identifiability constraints fix one class row and its bias to zero, or impose zero-sum constraints across the class rows and biases.

Normalisation layers

Batch normalisation, layer normalisation, instance normalisation, and group normalisation introduce additional symmetries in the network through their affine reparametrisation of activations. These layers were introduced and analysed in Ba et al. [69], Ioffe and Szegedy [817], Ulyanov et al. [1648], van Laarhoven [1665], with further work revealing implications for optimisation and generalisation. The precise symmetry depends both on *which* normalisation is used and on *which* statistics it employs (batch statistics during training versus stored running statistics at evaluation), so we state the assumptions explicitly.

Proposition 32.5 (BatchNorm Channel-scaling symmetry). *Consider a linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ followed by a batch-normalisation layer with learnable scale and shift $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^m$, using batch statistics and the idealized form without the numerical constant ϵ ,*

$$\text{norm}(\mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{\mathbf{z} - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta},$$

where $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}$ are the per-channel mean and standard deviation of $\mathbf{z} = \mathbf{W}x$. Let $\boldsymbol{\lambda} \in (\mathbb{R}_{>0})^m$ and $\mathbf{D} = \text{diag}(\boldsymbol{\lambda})$. Then, for all $x \in \mathbb{R}^n$,

$$\text{norm}(\mathbf{D}\mathbf{W}x, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \text{norm}(\mathbf{W}x, \boldsymbol{\beta}, \boldsymbol{\gamma}).$$

Proof. Let $\mathbf{z} = \mathbf{W}x \in \mathbb{R}^m$, and let $\tilde{\mathbf{z}} = \text{diag}(\boldsymbol{\lambda})\mathbf{z} = \text{diag}(\boldsymbol{\lambda})\mathbf{W}x$.

The normalisation of $\mathbf{z}$ is given by:

$$\text{norm}(\mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{\mathbf{z} - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta}.$$

Now consider the transformed input $\tilde{\mathbf{z}} = \text{diag}(\boldsymbol{\lambda})\mathbf{z}$. Then:

$$\text{norm}(\tilde{\mathbf{z}}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{\tilde{\mathbf{z}} - \tilde{\boldsymbol{\mu}}}{\tilde{\boldsymbol{\sigma}}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta},$$

where $\tilde{\boldsymbol{\mu}} = \text{diag}(\boldsymbol{\lambda})\hat{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}} = \text{diag}(\boldsymbol{\lambda})\hat{\boldsymbol{\sigma}}$, since both the mean and standard deviation scale linearly under element-wise positive scaling.

Therefore,

$$\begin{aligned} \text{norm}(\tilde{\mathbf{z}}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \frac{\text{diag}(\boldsymbol{\lambda})\mathbf{z} - \text{diag}(\boldsymbol{\lambda})\hat{\boldsymbol{\mu}}}{\text{diag}(\boldsymbol{\lambda})\hat{\boldsymbol{\sigma}}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta} \\ &= \frac{\text{diag}(\boldsymbol{\lambda})(\mathbf{z} - \hat{\boldsymbol{\mu}})}{\text{diag}(\boldsymbol{\lambda})\hat{\boldsymbol{\sigma}}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta} \\ &= \left(\frac{\mathbf{z} - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}} \right) \circ \boldsymbol{\gamma} + \boldsymbol{\beta} \\ &= \text{norm}(\mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\gamma}), \end{aligned}$$

which completes the proof. $\square$

Remark. With the ϵ added inside the square root, or with *stored* running statistics that do not co-scale with the activations, the cancellation is only approximate. The result also does not transfer unchanged to other normalisations: under LayerNorm the statistics are computed across the feature dimension, so an *arbitrary* coordinate-wise positive scaling does not cancel; under GroupNorm the invariance requires the scale λ to be constant within each normalised group.

Proposition 32.6 (Bias-shift symmetry of normalisation layers). *Consider a linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ followed by a normalisation layer with learnable scale and shift $\gamma, \beta \in \mathbb{R}^m$, and let $\mathbf{c} \in \mathbb{R}^m$ be a constant shift of the pre-normalisation activations $\mathbf{z} = \mathbf{W}x$. Two cases arise.*

1. **Centering with current activations.** *If the normalisation subtracts the mean of the current activations, then adding $\mathbf{c}$ shifts the mean by the same amount and is removed by the centering step. The output is invariant with β unchanged:*

$$\text{norm}(\mathbf{W}x + \mathbf{c}, \beta, \gamma) = \text{norm}(\mathbf{W}x, \beta, \gamma).$$

2. **Centering with stored statistics.** *If the normalisation uses fixed stored statistics $\hat{\mu}, \hat{\sigma}$ that do not move with $\mathbf{c}$, then the shift propagates through and must be absorbed by the shift parameter:*

$$\text{norm}(\mathbf{W}x + \mathbf{c}, \beta - \gamma \circ \mathbf{c} \oslash \hat{\sigma}, \gamma) = \text{norm}(\mathbf{W}x, \beta, \gamma),$$

where $\oslash$ denotes element-wise division.

In both cases the result follows from the fact that normalisation centres and rescales its input: a constant shift of the input is either annihilated by centering against the current mean (case 1) or exactly compensated by the shift parameter β (case 2). No scaling vector λ is involved in this symmetry.

Multi-head attention layer

Convention. Throughout this section, multi-head attention (MHA) uses h heads with $d = h d_h$ and projects an input sequence $x \in \mathbb{R}^{T \times d_{\text{model}}}$ by $\mathbf{Q} = x(\mathbf{W}^Q)^\top$, $\mathbf{K} = x(\mathbf{W}^K)^\top$, $\mathbf{V} = x(\mathbf{W}^V)^\top$, computes per-head attention, concatenates the heads, and applies the output projection $\mathbf{W}^O$. With this right-multiplication convention, block permutation and scaling matrices act on the *right* of the activations and on the *left* of the stored weight matrices.

Proposition 32.7 (Head Permutation Symmetries in MHA). *Let a MHA module use h attention heads, with query, key, and value weight matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_{\text{model}}}$, partitioned into h blocks along the output dimension. Let $\mathbf{W}^O \in \mathbb{R}^{d_{\text{model}} \times d}$ be the output projection matrix.*

Let $\pi \in P_h$ (note that $\pi \in \mathbb{R}^{h \times h}$) be a permutation matrix acting on the head dimension by permuting the order of the heads, and let $\Pi \in \mathbb{R}^{d \times d}$ be its block-wise extension acting on the concatenated output of all heads. Then, for all input sequences $x \in \mathbb{R}^{T \times d_{\text{model}}}$, the following transformation leaves the output of the MHA layer invariant:

$$\mathbf{W}^Q \mapsto \Pi \mathbf{W}^Q, \quad \mathbf{W}^K \mapsto \Pi \mathbf{W}^K, \quad \mathbf{W}^V \mapsto \Pi \mathbf{W}^V, \quad \mathbf{W}^O \mapsto \mathbf{W}^O \Pi^{-1}.$$

That is,

$$\text{MHA}(x; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O) = \text{MHA}(x; \Pi \mathbf{W}^Q, \Pi \mathbf{W}^K, \Pi \mathbf{W}^V, \mathbf{W}^O \Pi^{-1}).$$

Here $\Pi = \pi \otimes \mathbf{I}_{d_h}$ and $\Pi^{-1} = \pi^{-1} \otimes \mathbf{I}_{d_h}$ are the block-permutation matrix and its inverse acting on the concatenated heads, $\otimes$ is the Kronecker product, and $d_h = d/h$.

The proof of proposition 32.7 is left as an exercise for the reader.

Proposition 32.8 (Head-wise Query-Key Scaling Symmetry). *Let h be the number of attention heads, and $d = h \cdot d_h$ the total dimension of the concatenated heads. For any positive scaling vector $\alpha = (\alpha_1, \dots, \alpha_h) \in (\mathbb{R}_{>0})^h$, define the block-diagonal matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ as:*

$$\mathbf{A} = \text{diag}(\alpha_1 \mathbf{I}_{d_h}, \dots, \alpha_h \mathbf{I}_{d_h}),$$

and let $\mathbf{A}^{-1}$ be its inverse:

$$\mathbf{A}^{-1} = \text{diag}(\alpha_1^{-1} \mathbf{I}_{d_h}, \dots, \alpha_h^{-1} \mathbf{I}_{d_h}).$$

Then for all input sequences $x \in \mathbb{R}^{T \times d_{\text{model}}}$, the following transformation leaves the MHA output invariant:

$$\mathbf{W}^Q \mapsto \mathbf{A} \mathbf{W}^Q, \quad \mathbf{W}^K \mapsto \mathbf{A}^{-1} \mathbf{W}^K.$$

That is,

$$\text{MHA}(x; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O) = \text{MHA}(x; \mathbf{A} \mathbf{W}^Q, \mathbf{A}^{-1} \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O).$$

Table 32.1: Weight-space symmetries of common neural-network building blocks. Gaussian-prior invariance (PI) refers to an independent isotropic prior $\mathcal{N}(0, \sigma^2 I)$ on all weights and biases. Y : the prior measure is invariant under the transformation ($|\det J_{\mathcal{T}}| = 1$). N : the likelihood is invariant but the prior is not ($|\det J_{\mathcal{T}}| \neq 1$). m = hidden-layer width; L = number of hidden layers; K = number of output classes; $d_{\mathbf{h}}$ = penultimate feature dimension.

Symmetry	Assumptions	Group / transform.	PI	Inference consequence
<i>MLP hidden layers</i>				
Hidden-unit permutation	Any element-wise activation; nondegenerate units	$(\mathfrak{S}_m)^L$; finite, order $(m!)^L$; permutes rows of (W_ℓ, b_ℓ) and columns of $W_{\ell+1}$	Y	$(m!)^L$ identical posterior modes
ReLU positive scaling	Degree-1 pos. homogeneous activation; nondegenerate units	$(\mathbb{R}_{>0})^{mL}$; continuous Lie group, dim. mL ; scales rows of (W_ℓ, b_ℓ) by $\lambda_i > 0$ and columns of $W_{\ell+1}$ by λ_i^{-1}	N	Connected equal-loss ridges in the likelihood; posterior density varies along ridge under Gaussian prior
Odd-activation sign flip	Odd activation; nondegenerate units	$(\mathbb{Z}/2\mathbb{Z})^{mL}$; finite, order 2^{mL} ; negates row i of (W_ℓ, b_ℓ) and column i of $W_{\ell+1}$	Y	2^{mL} additional identical posterior copies
<i>Classification output layer</i>				
Softmax logit shift	K -class softmax output layer	$\mathbb{R}^{d_{\mathbf{h}}+1}$; continuous; $W \mapsto W + \mathbf{1}_K \mathbf{u}^\top$, $\mathbf{b} \mapsto \mathbf{b} + c \mathbf{1}_K$	N	$(d_{\mathbf{h}} + 1)$ -dimensional unidentifiable subspace in the final linear map
<i>Normalisation layers</i>				
BatchNorm channel scaling & bias shift	BatchNorm with learnable γ, β ; batch statistics; idealized ($\varepsilon = 0$)	$(\mathbb{R}_{>0})^m \ltimes \mathbb{R}^m$; continuous; scales and shifts pre-norm activations, absorbed by γ, β	N	Likelihood ridge; posterior density varies under Gaussian prior; cancellation approximate for $\varepsilon > 0$ or stored running statistics
LayerNorm & GroupNorm	GroupNorm with G groups; constant symmetry within each normalised group	$(\mathbb{R}_{>0})^G \ltimes \mathbb{R}^G$; continuous; group-restricted scaling and shift only	N	Narrower ridge than BatchNorm; arbitrary coordinate-wise scaling breaks LayerNorm invariance

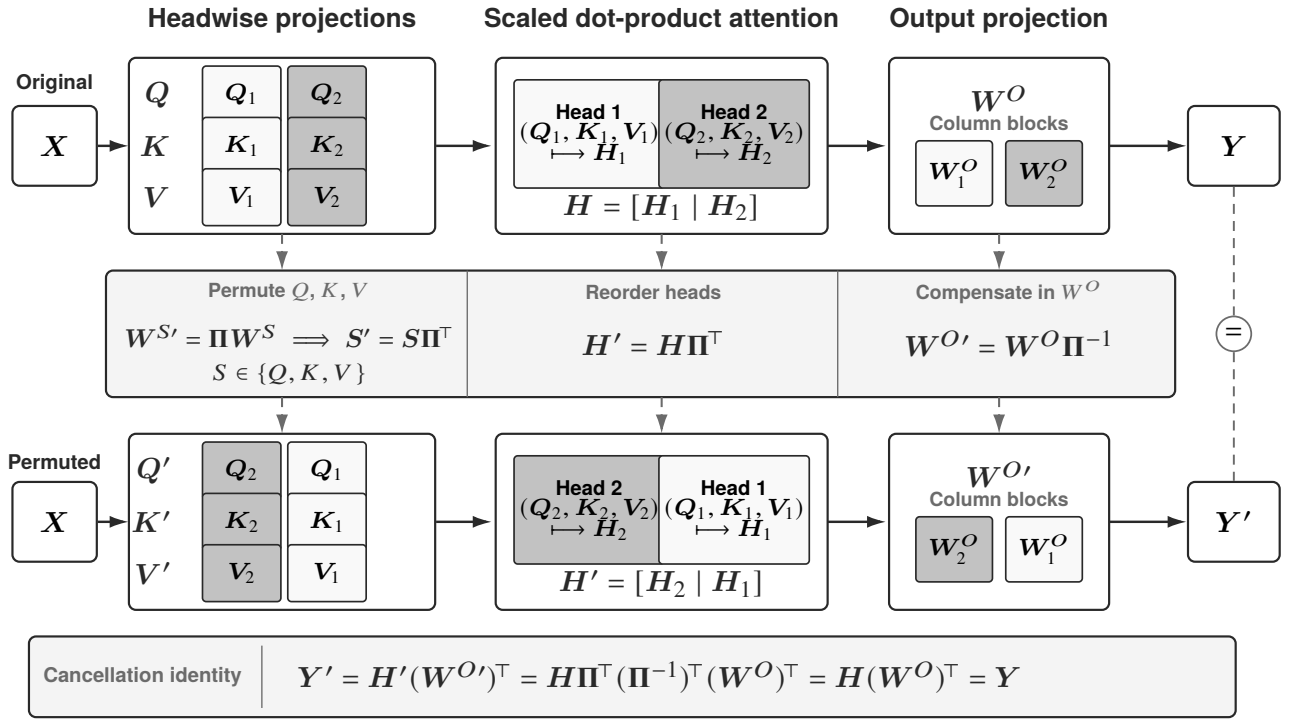


Figure 32.3: Two-head illustration of attention-head permutation symmetry. The block permutation $\Pi = \pi \otimes I_{d_h}$ permutes the row blocks of W^Q , W^K , and W^V , thereby reordering the corresponding query, key, value, and attention-head blocks. Applying the inverse permutation to the column blocks of the output projection, $W^{O'} = W^O\Pi^{-1}$, exactly cancels this reordering, so that the MHA output is unchanged: $Y' = Y$. For the two-head transposition illustrated here, $\Pi^{-1} = \Pi$.

Proof. We analyse the effect of the proposed transformation on the MHA computation. The MHA mechanism proceeds as follows:

1. Compute the query, key, and value projections:

$$Q = xW^{Q^\top}, \quad K = xW^{K^\top}, \quad V = xW^{V^\top}.$$

Each of these has shape $T \times d$, with $d = h \cdot d_h$, and can be split into h blocks of size d_h per head.

2. Apply attention separately for each head $i = 1, \dots, h$:

$$\text{head}_i = \text{Softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_h}}\right) V_i,$$

where $Q_i, K_i, V_i \in \mathbb{R}^{T \times d_h}$ are the corresponding blocks for head i .

3. Concatenate the heads and project with W^O :

$$\text{MHA}(x) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot W^{O^\top}.$$

Now consider modifying W^Q and W^K using the block-diagonal scaling matrix A and its inverse:

$$W_{\text{new}}^Q = AW^Q, \quad W_{\text{new}}^K = A^{-1}W^K.$$

Then:

$$Q_{\text{new}} = xW^{Q^\top}A^\top, \quad K_{\text{new}} = xW^{K^\top}(A^{-1})^\top.$$

Hence the attention score matrix becomes:

$$\frac{Q_{\text{new}}K_{\text{new}}^\top}{\sqrt{d_h}} = \frac{(xW^{Q^\top}A^\top)(xW^{K^\top}(A^{-1})^\top)^\top}{\sqrt{d_h}}.$$

Since $\mathbf{A}$ is diagonal, it is symmetric ($\mathbf{A} = \mathbf{A}^\top$), and the central factor satisfies $\mathbf{A}^\top ((\mathbf{A}^{-1})^\top)^\top = \mathbf{A} \mathbf{A}^{-1} = \mathbf{I}$. Thus,

$$\frac{\mathbf{Q}_{\text{new}} \mathbf{K}_{\text{new}}^\top}{\sqrt{d_h}} = \frac{\mathbf{Q} \mathbf{K}^\top}{\sqrt{d_h}},$$

which implies:

$$\text{Softmax} \left(\frac{\mathbf{Q}_{\text{new}} \mathbf{K}_{\text{new}}^\top}{\sqrt{d_h}} \right) = \text{Softmax} \left(\frac{\mathbf{Q} \mathbf{K}^\top}{\sqrt{d_h}} \right).$$

As a result, the attention weights for each head are unchanged, and the resulting attention outputs are the same for each head. Since the values $\mathbf{V}$ and output projection $\mathbf{W}^O$ are unchanged, the final MHA output remains invariant. $\square$

Proposition 32.9 (Head-wise Value Scaling Symmetry). *Let h be the number of attention heads and $d = h \cdot d_h$ the total concatenated head dimension. For any invertible scaling vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_h) \in (\mathbb{R} \setminus \{0\})^h$, define the block-diagonal matrix $\boldsymbol{\Lambda} = \text{diag}(\lambda_1 \mathbf{I}_{d_h}, \dots, \lambda_h \mathbf{I}_{d_h}) \in \mathbb{R}^{d \times d}$ and its inverse $\boldsymbol{\Lambda}^{-1} = \text{diag}(\lambda_1^{-1} \mathbf{I}_{d_h}, \dots, \lambda_h^{-1} \mathbf{I}_{d_h})$.*

Then for all input sequences $x \in \mathbb{R}^{T \times d_{\text{model}}}$, the following transformation leaves the MHA output invariant:

$$\mathbf{W}^V \mapsto \boldsymbol{\Lambda} \mathbf{W}^V, \quad \mathbf{W}^O \mapsto \mathbf{W}^O \boldsymbol{\Lambda}^{-1}.$$

That is,

$$\text{MHA}(x; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O) = \text{MHA}(x; \mathbf{W}^Q, \mathbf{W}^K, \boldsymbol{\Lambda} \mathbf{W}^V, \mathbf{W}^O \boldsymbol{\Lambda}^{-1}).$$

Proof. Let us examine the MHA computation steps. For an input $x \in \mathbb{R}^{T \times d_{\text{model}}}$:

- Compute projections:

$$\mathbf{Q} = x \mathbf{W}^{Q^\top}, \quad \mathbf{K} = x \mathbf{W}^{K^\top}, \quad \mathbf{V} = x \mathbf{W}^{V^\top} \in \mathbb{R}^{T \times d}.$$

- Split $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ into heads:

$$\mathbf{Q} = [\mathbf{Q}_1 \parallel \dots \parallel \mathbf{Q}_h], \quad \mathbf{K} = [\mathbf{K}_1 \parallel \dots \parallel \mathbf{K}_h], \quad \mathbf{V} = [\mathbf{V}_1 \parallel \dots \parallel \mathbf{V}_h],$$

where $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{T \times d_h}$.

- Each attention head is:

$$\text{head}_i = \text{Softmax} \left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_h}} \right) \mathbf{V}_i.$$

- Final output:

$$\text{MHA}(x) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot \mathbf{W}^{O^\top}.$$

Now apply the transformation $\mathbf{W}^V \mapsto \boldsymbol{\Lambda} \mathbf{W}^V$, so that:

$$\mathbf{V}' = x \mathbf{W}^{V^\top} \boldsymbol{\Lambda}^\top = \mathbf{V} \boldsymbol{\Lambda}^\top.$$

Because $\boldsymbol{\Lambda}$ is block-diagonal, this scales each head's values:

$$\mathbf{V}'_i = \lambda_i \mathbf{V}_i \quad \text{for each } i.$$

Therefore, the new attention heads are:

$$\text{head}'_i = \text{Softmax} \left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_h}} \right) \cdot \lambda_i \mathbf{V}_i = \lambda_i \cdot \text{head}_i.$$

Now concatenate:

$$\text{Concat}(\text{head}'_1, \dots, \text{head}'_h) = \text{Concat}(\lambda_1 \text{head}_1, \dots, \lambda_h \text{head}_h) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot \boldsymbol{\Lambda}^\top.$$

Finally, apply the transformed output projection:

$$\text{MHA}'(x) = \text{Concat}(\text{head}'_1, \dots, \text{head}'_h) \cdot (\mathbf{W}^O \boldsymbol{\Lambda}^{-1})^\top.$$

Substitute the concatenated form:

$$= (\text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot \boldsymbol{\Lambda}^\top) \cdot (\boldsymbol{\Lambda}^{-1})^\top \mathbf{W}^{O^\top}.$$

But since $\boldsymbol{\Lambda}^\top (\boldsymbol{\Lambda}^{-1})^\top = \mathbf{I}$, we have:

$$\text{MHA}'(x) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot \mathbf{W}^{O^\top} = \text{MHA}(x).$$

$\square$

We summarise the MHA-related symmetries in table 32.2.

Table 32.2: Weight-space symmetries of common transformers building blocks. Gaussian-prior invariance (PI) refers to an independent isotropic prior $\mathcal{N}(0, \sigma^2 I)$ on all weights and biases. Y : the prior measure is invariant under the transformation. N : the likelihood is invariant but the prior is not. m = hidden-layer width; h = number of attention heads; $d_h = d/h$ = per-head dimension; K = number of output classes.

Symmetry	Assumptions	Group / transform.	PI	Inference consequence
<i>Multi-head attention</i>				
Attention-head permutation	h equal-size d_h -dim heads; MHA with W^Q, W^K, W^V, W^O	$\mathfrak{S}_h$; finite, order $h!$; block-permutes rows of (W^Q, W^K, W^V) and columns of W^O via $\Pi = \pi \otimes I_{d_h}$	Y	$h!$ equivalent configurations per attention layer; compounds with hidden-unit symmetry in transformer blocks
QK inverse scaling	h -head MHA; per-head query and key projections	$(\mathbb{R}_{>0})^h$; continuous; $W_i^Q \mapsto \alpha_i W_i^Q$, $W_i^K \mapsto \alpha_i^{-1} W_i^K$ per head i	N	Equal attention-score manifold per head; $\ W_i^Q\ _F$ and $\ W_i^K\ _F$ change asymmetrically; posterior varies
Value-output inverse scaling	h -head MHA; invertible per-head scaling ($\lambda_i \neq 0$)	$(\mathbb{R} \setminus \{0\})^h$; continuous; $W_i^V \mapsto \lambda_i W_i^V$, $W_i^O \mapsto \lambda_i^{-1} W_i^O$ per head i	N	Connected equal-output manifold; more general than ReLU scaling (allows sign flips)

32.3 From likelihood symmetries to posterior symmetries

The previous section catalogued transformations $\mathcal{T}$ that leave the implemented function unchanged, $f_{\mathcal{T}(\theta)} = f_\theta$. We now make explicit what they do, and do *not*, imply for Bayesian inference. This is the central point for Bayesian deep learning: a function symmetry is automatically a *likelihood* symmetry, but it is a *posterior* symmetry only under an additional condition on the prior. Tables 32.1 and 32.2 include a column indicating whether the usual Gaussian prior is also invariant under the transformation.

Likelihood invariance. Because the likelihood depends on θ only through the function f_θ , any weight-space symmetry preserves it:

$$p(\mathcal{D} \mid \mathcal{T}(\theta)) = p(\mathcal{D} \mid \theta), \quad \text{since } f_{\mathcal{T}(\theta)} = f_\theta.$$

Equivalently, the likelihood is constant on every orbit $G \cdot \theta$.

When is the posterior invariant? Write the posterior density (with respect to a chosen base measure on $\mathbb{R}^d$, typically Lebesgue) as $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta) \rho(\theta)$, where ρ is the prior density. For an invertible, differentiable transformation $\mathcal{T}$ with Jacobian $J_\mathcal{T}$, the pushforward of the posterior under $\mathcal{T}$ has density proportional to $p(\mathcal{D} \mid \mathcal{T}^{-1}(\theta)) \rho(\mathcal{T}^{-1}(\theta)) \mid \det J_{\mathcal{T}^{-1}}(\theta) \mid$. Using likelihood invariance, the posterior is invariant under $\mathcal{T}$ if and only if the prior *measure* is invariant, i.e.

$$\rho(\mathcal{T}(\theta)) \mid \det J_\mathcal{T}(\theta) \mid = \rho(\theta) \quad \text{for all } \theta. \quad (32.2)$$

The factor $\mid \det J_\mathcal{T} \mid$ is exactly the change in the base measure; ignoring it is a common source of confusion. A function symmetry that violates (32.2) still leaves the likelihood flat along the orbit, but the posterior density *varies* along it.

Permutations are posterior symmetries (under exchangeable priors). A hidden-unit permutation acts by an orthogonal coordinate permutation, so $\mid \det J_\mathcal{T} \mid = 1$. If the prior is exchangeable, i.e. the per-unit incoming and outgoing weights and biases are independent and identically distributed across the m hidden units, then ρ is permutation-invariant and (32.2) holds. Permutations are therefore genuine posterior symmetries: the posterior has, up to stabilizers and degenerate units, $(m!)^L$ identical copies, one per ordering. Canonical ordering of units can remove these copies.

ReLU scaling is usually *not* a posterior symmetry. The scaling transformation $W_1 \mapsto \text{diag}(\lambda) W_1$, $b_1 \mapsto \text{diag}(\lambda) b_1$, $W_2 \mapsto W_2 \text{diag}(\lambda)^{-1}$ is linear in θ with

$$\mid \det J_\mathcal{T} \mid = \prod_{i=1}^m \lambda_i^{(n+1)-k},$$

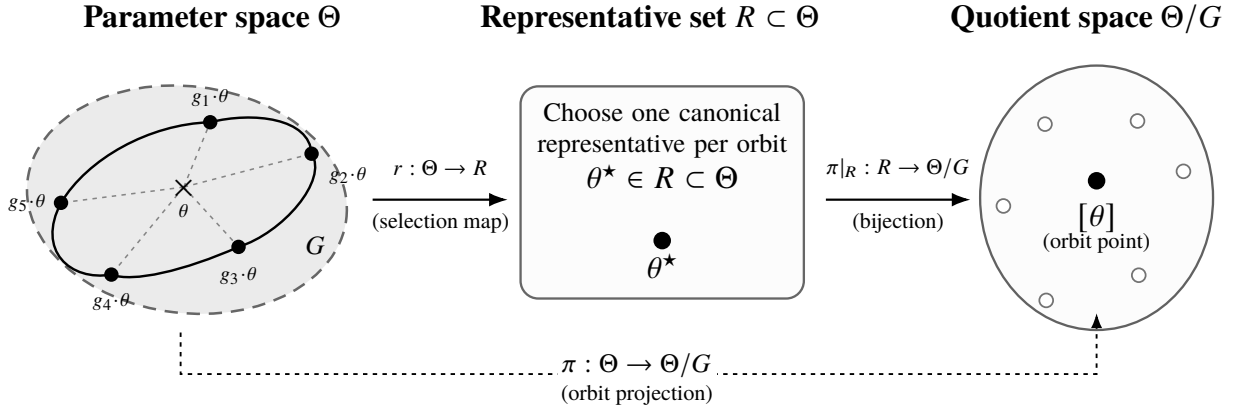


Figure 32.4: Quotienting parameter space by a symmetry group. The orbit $\mathcal{O}_\theta = \{g \cdot \theta \mid g \in G\}$ contains parameter vectors related by the action of G . A representative-selection map $r: \Theta \rightarrow R$ chooses a canonical parameter θ^* from each orbit, where $R \subset \Theta$ contains one representative per orbit. The quotient projection $\pi: \Theta \rightarrow \Theta/G$ maps every parameter vector in the same orbit to the common equivalence class $[\theta]$. Consequently, the restriction $\pi|_R: R \rightarrow \Theta/G$ is a bijection, so the representative set R provides a concrete realisation of the quotient space.

since each of the $n + 1$ entries of row i of $(\mathbf{W}_1, \mathbf{b}_1)$ is multiplied by λ_i and each of the k entries of column i of $\mathbf{W}_2$ by λ_i^{-1} . Under an independent Gaussian prior (equivalently, ℓ_2 weight decay), $\rho(\mathcal{T}(\theta)) \neq \rho(\theta)$ in general, and the factor above does not restore (32.2). Hence ReLU scaling preserves the likelihood, producing a flat *ridge*, but tilts the posterior density along that ridge: it is a likelihood symmetry, not a posterior symmetry. It becomes a posterior symmetry only for a scale-invariant prior chosen so that (32.2) holds.

Inference on the quotient. When a transformation *is* a genuine posterior symmetry, the redundant copies carry no extra information and can be removed by restricting inference to a symmetry-free reference set, one representative per orbit, equipped with the correct quotient measure (the pushforward of the prior under the canonicalisation map). Sampling or variational inference on this quotient avoids revisiting equivalent modes and yields better-calibrated diagnostics, while leaving the predictive distribution in function space unchanged. The practical procedures of the next section construct such representatives.

32.4 Choosing a representative modulo symmetries

Weight-space symmetries are transformations of the parameters that leave the function implemented by the network unchanged. Although they do not affect the output, they introduce redundancy into the parameter space, which could be problematic for several tasks:

- **Visualisation and interpretation of parameter space:** symmetries can distort geometry, making it hard to interpret loss landscapes [826, 980].
- **Bayesian inference:** redundant modes can slow down or confuse posterior sampling in Bayesian neural networks [980, 1029].
- **Weight averaging and ensembles:** permutation symmetries explain why *naive* parameter averaging can fail, because matched units may appear in different orders in two independently trained networks. Aligning the networks modulo their symmetries before averaging is what makes the operation meaningful, as in *Git Re-Basin* [15].

We select one representative from each orbit, i.e. a canonicalisation of the quotient $\mathbb{R}^d/G$ of section 32.1. In this section we describe practical procedures for two common symmetries in two-layer ReLU MLPs, *scaling* and *permutation* of the hidden units. We assume throughout that no hidden unit is degenerate (no zero row of $\mathbf{W}_1$ and no zero column of $\mathbf{W}_2$); degenerate units have to be handled separately because they break the one-to-one correspondence between units and orbit coordinates.

Discrete versus continuous symmetries. The two symmetries are of fundamentally different types, and they require different tools. For an MLP with L hidden layers of width m , the permutations form a *finite* group of size $(m!)^L$, whereas the ReLU positive scalings form a *continuous* Lie group $(\mathbb{R}_{>0})^{mL}$ of dimension mL . Under exchangeable priors, the discrete symmetries produce a finite set of equivalent posterior modes, for which a canonical ordering of the units is a natural representative. The continuous likelihood symmetries instead produce positive-dimensional loss orbits, and yield posterior *ridges* only when the prior and base measure also preserve the symmetry; here a representative constraint or an equi-normalisation is the appropriate tool. We treat the two cases in turn and then combine them. In the Bayesian setting, canonical ordering of hidden units has been used to improve BNN inference [1342], and symmetry-free reference sets together with symmetry-aware MCMC have been proposed to sample the quotient directly [1747].

32.4.1 Scaling symmetry and its representative

As described in proposition 32.3, if r is positively homogeneous of degree one, then for any positive vector $\lambda \in (\mathbb{R}_{>0})^m$ and $D = \text{diag}(\lambda)$,

$$W_2 r(W_1 x + b_1) + b_2 = W_2 D^{-1} r(D W_1 x + D b_1) + b_2.$$

Scaling row i of (W_1, b_1) by λ_i and column i of W_2 by λ_i^{-1} leaves the output unchanged. To select a representative, we fix a canonical scale by normalising each row of W_1 (including the corresponding entry of b_1) to unit ℓ_2 norm and compensating in W_2 .

Pseudocode for removing scaling symmetry:

```

1 def canonical_scaling(W1, W2):
2     # W1: weight matrix from input to hidden (m x n)
3     # W2: weight matrix from hidden to output (k x m)
4
5     m = W1.shape[0] # number of hidden units
6     eps = 1e-8      # small constant to avoid division by zero
7
8     # Step 1: Compute L2 norm of each row of W1
9     lambdas = np.linalg.norm(W1, axis=1) + eps # shape (m,)
10
11    # Step 2: Normalise W1 rows
12    W1_canon = W1 / lambdas[:, None]           # shape (m x n)
13
14    # Step 3: Rescale W2 columns
15    W2_canon = W2 * lambdas[None, :]           # shape (k x m)
16
17    return W1_canon, W2_canon

```

Listing 32.1: Fix the scaling representative for a 2-layer MLP

This procedure maps all parameter configurations in the same scaling orbit to a single representative in which every row of W_1 has unit norm. The degree-one positive homogeneity of the activation guarantees that the network output is unchanged, since the compensating rescaling of W_2 exactly cancels the rescaling of W_1 . The ℓ_2 norm is the standard choice, but other norms such as ℓ_1 or the max-norm can be used.

32.4.2 Permutation symmetry and its representative

As described in proposition 32.2, permuting the order of the hidden units does not affect the function computed by the network:

$$W_2 r(W_1 x + b_1) + b_2 = W_2 \pi^{-1} r(\pi W_1 x + \pi b_1) + b_2,$$

where π is a permutation matrix. To select a representative, we sort the hidden units by a fixed scalar score and reorder the rows of (W_1, b_1) and the columns of W_2 accordingly.

Choice of an invariant score. A subtle point is that the score must be a genuine invariant of the orbit we still wish to quotient. Sorting by the raw row norm $\|W_1[i]\|$ is *not* valid in the presence of scaling symmetry: two parameters in the same scaling orbit can have different row norms and hence different sort orders, so the

representative would not be well defined. We therefore fix the scale first (previous subsection) and then sort by a quantity that is invariant under the residual symmetry. After normalising the rows of $\mathbf{W}_1$ to unit norm, all of them have score 1, so the row norm is uninformative and any tie-break would come from the arbitrary $\mathbf{eps}$ rather than from the data. A meaningful choice is the *outgoing* weight magnitude, the norm of column i of $\mathbf{W}_2$ after normalisation, which equals the product of incoming and outgoing norms in the original parametrisation and is therefore invariant along the scaling orbit. The pseudocode below assumes the rows of $\mathbf{W}_1$ have already been normalised.

Pseudocode for removing permutation symmetry:

```

1 def canonical_permutation(W1, W2):
2     # Assumes rows of W1 are already L2-normalised (scale fixed).
3     # W1: input-to-hidden weights (m x n)
4     # W2: hidden-to-output weights (k x m)
5
6     # Step 1: Score each hidden unit by a scaling-invariant quantity:
7     #         the outgoing-weight norm (column norm of W2).
8     scores = np.linalg.norm(W2, axis=0) # shape (m,)
9
10    # Step 2: Sort hidden units by descending score
11    perm = np.argsort(-scores)          # permutation indices
12
13    # Step 3: Apply the permutation consistently
14    W1_canon = W1[perm, :]              # permute rows
15    W2_canon = W2[:, perm]              # permute columns
16
17    return W1_canon, W2_canon

```

Listing 32.2: Choose a permutation representative for a scale-normalised 2-layer MLP

Sorting by a scaling-invariant outgoing-weight norm fixes a canonical order of the hidden units, so that all parameter configurations related by a permutation (after the scale has been fixed) map to a single representative. This is particularly useful when comparing or averaging models, visualizing learning dynamics, or interpreting individual neurons. When units are nearly tied, an alternative to a brittle scalar sort is to *align* the units to those of a fixed reference model by solving a matching problem (a linear assignment), as done in model-merging methods.

32.4.3 Combining scaling and permutation representatives

To select a joint representative for both symmetries, we first fix the scale by normalising the hidden-unit rows, and then sort the units by the scaling-invariant criterion above. Because the sort key is chosen to be invariant under the scaling that we have already fixed, the order of the two operations is well defined and the composition yields a unique, well-defined representative of the parameter orbit for a given network function. This is the canonicalisation corresponding to the joint group of permutations and positive scalings. For deeper networks, the scale-fixing step generalises to multi-layer *equi-normalisation*, which balances the scales across all layers [1564].

32.5 Mode connectivity

The representatives of the previous section are useful precisely because the loss landscape is shaped by symmetries. Two networks that are functionally equivalent, or merely similar, may sit far apart in parameter space simply because their hidden units are permuted or rescaled. *Mode connectivity* studies whether two trained solutions can be joined by a path of low loss, and the symmetry viewpoint explains much of what is observed: aligning one network to another modulo permutation symmetry often turns an apparent high-loss interpolation into a near-flat one. Beyond the linear case considered here, Draxler et al. [451] showed that independently trained minima are typically connected by *nonlinear* low-loss curves, and Brea et al. [204] analysed how permutation symmetries organize the minima into connected structures. We make these notions precise and then revisit alignment.

Let $\Theta \subset \mathbb{R}^d$ be the parameter space of a DNN. A DNN is a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, parameterised by weights $\theta \in \Theta$. Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$, the network is trained to minimise the empirical loss

$$\mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x_i), y_i),$$

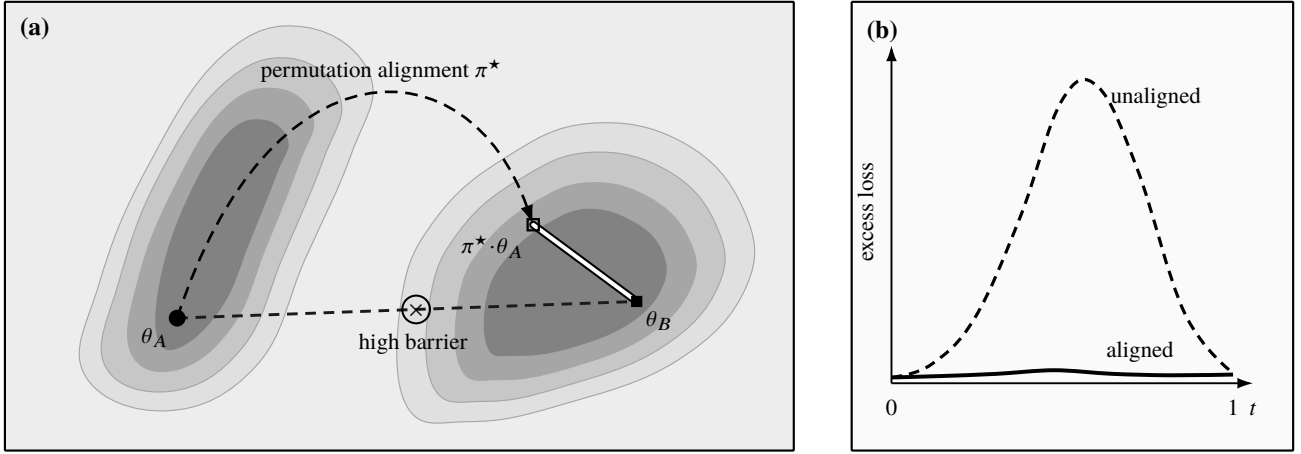


Figure 32.5: **Permutation alignment reveals linear mode connectivity.** (a) Two independently trained solutions θ_A and θ_B lie in different low-loss basins, so their naive linear interpolation (dashed) crosses a high-loss region. A hidden-unit permutation π^* maps θ_A to the functionally equivalent representative $\pi^* \cdot \theta_A$ in the basin of θ_B ; the aligned interpolation (solid) then remains at low loss. (b) Schematic excess loss along the two segments: the unaligned interpolation has a pronounced barrier, whereas the aligned interpolation is nearly flat.

where ℓ is a task-dependent loss function (e.g., cross-entropy). We do not assume $\mathcal{L} \geq 0$: for instance a Gaussian negative log-likelihood with small variance can be negative.

Definition 32.13 (Low-loss set). Let $L_{\min} = \inf_{\theta \in \Theta} \mathcal{L}(\theta)$ and fix a tolerance $\epsilon > 0$. The low-loss set is

$$S_\epsilon := \{\theta \in \Theta \mid \mathcal{L}(\theta) \leq L_{\min} + \epsilon\},$$

the set of parameters whose loss is within ϵ of the best attainable value.

Definition 32.14 (Loss Barrier [493, 554]). Let $\theta_A, \theta_B \in \Theta$. Define the loss barrier between θ_A and θ_B as

$$\mathcal{B}(\theta_A, \theta_B) := \max_{t \in [0,1]} \tilde{\mathcal{L}}_t(\theta_A, \theta_B),$$

where

$$\tilde{\mathcal{L}}_t(\theta_A, \theta_B) := \mathcal{L}((1-t)\theta_A + t\theta_B) - ((1-t)\mathcal{L}(\theta_A) + t\mathcal{L}(\theta_B)).$$

The loss barrier measures the excess loss along the linear interpolation between θ_A and θ_B , relative to the linear interpolation of endpoint losses.

Definition 32.15 (Linearly Mode-Connected Solutions). Two solutions $\theta_A, \theta_B \in S_\epsilon$ with approximately equal losses are linearly mode-connected (LMC) if the linear interpolation between them stays at near-constant loss, i.e., $\mathcal{B}(\theta_A, \theta_B) \approx 0$.

Symmetry and the loss barrier. It is tempting to state a proposition asserting that two functionally equivalent networks have a zero loss barrier. As written, such a statement is a tautology: if $\mathcal{T}_*(\theta_A) = \theta_B$ for a function-preserving $\mathcal{T}_*$, then the linear interpolation between $\mathcal{T}_*(\theta_A)$ and θ_B is the constant path at θ_B , so the barrier vanishes by construction and nothing has been learned about the landscape. The substantive content is empirical and concerns *distinct* parameter vectors.

Empirical observation (permutation alignment reduces the barrier) [15, 493]. Let θ_A and θ_B be two networks of the same architecture trained *independently* (different random seeds), so that in general $\theta_A \neq \theta_B$ and the naive linear interpolation between them crosses a high-loss region. Entezari et al. conjectured, and Ainsworth et al. gave a practical method (Git Re-Basin) showing, that there is often a permutation π^* of the hidden units such that the *aligned* network $\pi^* \cdot \theta_A$ and the reference θ_B have a near-zero linear loss barrier:

$$\mathcal{B}(\pi^* \cdot \theta_A, \theta_B) \approx 0, \quad \text{while } \pi^* \cdot \theta_A \neq \theta_B.$$

This is an empirical observation about independently trained solutions: the two aligned parameter vectors remain distinct, yet the path between them stays at low loss in their experiments.

The practical reading is that an *apparent* positive barrier between two good solutions is often, at least in part, an artifact of permutation misalignment in parameter space. Removing that misalignment by aligning the networks modulo permutation symmetry frequently reveals an underlying low-loss connection.

Definition 32.16 (Local Convexity). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable in a neighbourhood of a point $\bar{x} \in \mathbb{R}^n$. We say that f is locally convex at $\bar{x}$ if its Hessian matrix*

$$\nabla^2 f(\bar{x}) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(\bar{x}) \right]_{i,j=1}^n$$

is positive semidefinite, that is, $d^\top \nabla^2 f(\bar{x}) d \geq 0$ for all $d \in \mathbb{R}^n$.

We use positive semi-definiteness rather than strict positive definiteness. In any model with continuous likelihood symmetries (ReLU positive scaling, BatchNorm channel scalings, and similar), the Hessian has zero eigenvalues in the directions tangent to the symmetry orbit, so strict positive definiteness is unavailable. Positive semidefiniteness keeps the convexity intuition while remaining compatible with these flat symmetry directions.

Definition 32.17 (Locally convex region). *Let $\mathcal{L} : \Theta \rightarrow \mathbb{R}$ be twice continuously differentiable. A set $C \subset \Theta$ is called a locally convex region if*

$$d^\top \nabla^2 \mathcal{L}(\theta) d \geq 0 \quad \text{for all } \theta \in C \text{ and all } d \in \mathbb{R}^d,$$

i.e. $\nabla^2 \mathcal{L}(\theta) \succeq 0$ throughout C .

Proposition 32.10 (Convexity inequality along segments inside a locally convex region). *Let C be a locally convex region as defined in Definition 32.17. If $\theta_A, \theta_B \in C$ and the whole line segment*

$$[\theta_A, \theta_B] := \{(1-t)\theta_A + t\theta_B : t \in [0, 1]\}$$

is contained in C , then for all $t \in [0, 1]$,

$$\mathcal{L}((1-t)\theta_A + t\theta_B) \leq (1-t)\mathcal{L}(\theta_A) + t\mathcal{L}(\theta_B) \leq \max\{\mathcal{L}(\theta_A), \mathcal{L}(\theta_B)\}.$$

In particular, the excess loss $\tilde{\mathcal{L}}_t(\theta_A, \theta_B)$ is nonpositive for every t , so the linear loss barrier along the segment satisfies $\mathcal{B}(\theta_A, \theta_B) \leq 0$: there is no barrier between θ_A and θ_B along this segment.

Proof. Define the one-dimensional restriction $g(t) := \mathcal{L}((1-t)\theta_A + t\theta_B)$ for $t \in [0, 1]$. Then g is twice differentiable and

$$g''(t) = (\theta_B - \theta_A)^\top \nabla^2 \mathcal{L}((1-t)\theta_A + t\theta_B) (\theta_B - \theta_A) \geq 0$$

because the segment lies in C and $\nabla^2 \mathcal{L} \succeq 0$ on C . Hence g is convex on $[0, 1]$, so $g(t) \leq (1-t)g(0) + tg(1)$. This yields the first inequality; the second is a basic bound for convex combinations. $\square$

Proposition 32.11 (Weight averaging vs. output averaging, to first order [824]). *Let $f_\theta : \mathcal{X} \rightarrow \mathbb{R}$ denote a (scalar) twice continuously differentiable network output with respect to parameters $\theta \in \mathbb{R}^d$. Let $\{\theta_i\}_{i=1}^n \subset C$ lie in a region C where a local neural tangent kernel/linearisation holds, i.e., $f_\theta(x)$ is locally affine in θ . Define the Stochastic*

Weight average (SWA) $\theta_{\text{SWA}} := \frac{1}{n} \sum_{i=1}^n \theta_i$ and deviations $\Delta_i := \theta_i - \theta_{\text{SWA}}$ (so $\sum_i \Delta_i = 0$). Then, for every $x \in \mathcal{X}$,

$$f_{\theta_{\text{SWA}}}(x) = \frac{1}{n} \sum_{i=1}^n f_{\theta_i}(x) + O(\Delta^2), \quad \text{where } \Delta := \max_i \|\Delta_i\|.$$

Proof. By first-order Taylor expansion at θ_{SWA} ,

$$f_{\theta_i}(x) = f_{\theta_{\text{SWA}}}(x) + \langle \nabla_\theta f_{\theta_{\text{SWA}}}(x), \Delta_i \rangle + O(\|\Delta_i\|^2).$$

Averaging over i gives

$$\frac{1}{n} \sum_{i=1}^n f_{\theta_i}(x) = f_{\theta_{\text{SWA}}}(x) + \left\langle \nabla_\theta f_{\theta_{\text{SWA}}}(x), \frac{1}{n} \sum_{i=1}^n \Delta_i \right\rangle + O(\Delta^2) = f_{\theta_{\text{SWA}}}(x) + O(\Delta^2),$$

since $\sum_i \Delta_i = 0$. If f_θ is globally affine in θ , all second-order terms vanish and the equality is exact. $\square$

Remark (scope of the result). Proposition 32.11 is a *local, first-order* statement about a scalar network output $f_{\theta}(x)$. It says that, near a common point and to first order, averaging the weights approximates averaging the outputs. It should not be read as an unconditional equivalence between SWA and a deep ensemble. In classification, for example, the predictive distribution applies a softmax to the logits, and averaging logits (which is what a first-order argument on the pre-softmax output gives) is *not* the same as averaging the predicted probabilities of the ensemble members. The two coincide only when the second-order terms are negligible and the read-out is linear; in general the ensemble of probabilities and the single averaged-weight model differ.

Intuitively, the set of weights $\{\theta_i\}_i$ used for weight averaging are sampled along a single training trajectory within a connected basin, where the loss surface is approximately flat or convex. Along such paths, the loss barrier between endpoints is essentially zero. By proposition 32.10, this implies that the loss of the averaged model θ_{SWA} is no worse (and typically better) than the average loss of the individual models with weights $\{\theta_i\}_i$.

This idea was first exploited in *Fast Geometric Ensembling* (FGE) [581], then formalised in *Stochastic Weight Averaging* (SWA) [824], and extended in *SWAG* [1116] and *Tracking of Deep neural network weight Distributions* (TRADI) [553], which leverage weight averaging as a cheap alternative to variational inference for posterior approximation. More recently, *Model Soup* [1776] and *Model Ratatouille* [1374] average weights from models sharing the same initialisation but trained under different fine-tuning strategies. Their approach ensures that the weights $\{\theta_i\}_i$ lie in regions connected by low-loss paths, thus maintaining small loss barriers across models.

Beyond averaging along a single trajectory, methods such as *Git Re-Basin* [15], *ZipIt* [1566], and *Starlight* [1544] merge *independently* trained models. The key step is to align the models modulo permutation symmetries before averaging or merging: by matching the hidden units of one network to those of the other, the two solutions are brought into a common, comparable parametrisation in which their interpolation has a small loss barrier, which is what makes the combination effective.

32.6 Conclusion

Symmetries are a fundamental aspect of neural network parameterisations, arising whenever distinct parameter configurations represent the same predictive function. Throughout this chapter, we have shown that these symmetries naturally emerge from common architectural components, including hidden-unit permutations, positive scaling transformations in homogeneous networks, normalisation layers, and multi-head attention. Viewing these transformations through the lens of group actions provides a unified framework for understanding the geometry of neural network parameter spaces and the resulting equivalence classes of models.

From a Bayesian perspective, these symmetries have important implications for posterior inference. Function-preserving transformations always induce invariances of the likelihood, whereas posterior invariance additionally depends on the prior respecting the same symmetry. Consequently, parameter symmetries can lead to multiple equivalent posterior modes or continuous low-probability valleys in weight space, complicating posterior exploration, variational approximations, and model comparison. Recognizing these structures is therefore essential for interpreting Bayesian uncertainty in parameter space and for designing inference algorithms that efficiently explore the posterior distribution.

We also discussed practical strategies for handling symmetries, including canonical representatives, symmetry-aware alignment, and parameter transformations prior to averaging or merging models. Rather than eliminating symmetries, these approaches operate on equivalence classes of parameters, allowing optimisation and inference procedures to avoid redundant exploration of identical solutions. This perspective also explains recent advances in mode connectivity, Git Re-Basin, and weight-space model merging, where aligning networks modulo their symmetries reveals unexpectedly simple geometric relationships between independently trained solutions.

More broadly, understanding neural network symmetries bridges abstract mathematical concepts such as group actions and quotient spaces with practical challenges in Bayesian deep learning. As models continue to grow in size and architectural complexity, symmetry-aware optimisation and inference are likely to become increasingly important, both for improving computational efficiency and for developing a more principled understanding of posterior geometry in overparameterised neural networks.

Chapter 33

Symmetries and sampling

The implications of weight-space symmetries in Bayesian neural networks on sampling-based inference are not straightforward to assess. This chapter discusses the influencing factors for how benign or adverse the posterior density presents itself to sampling, identifying a trade-off between symmetric structures and the degree of regularisation. Besides affecting the design of the sampling procedure, symmetries can result in misleading convergence diagnostics.

33.1 Posterior Landscape in the Presence of Symmetries

33.1.1 Posterior Landscape

Assessing the difficulty to be expected for a sampler traversing the posterior state space requires the consideration of both *topological* and *geometrical* aspects. The former pertains to the number and relative location of *modes*, while the latter concerns questions of *curvature*. Together, these properties characterise the posterior *landscape*. Behind this lies a consideration of posterior *mass* (volume times density): Bayesian model averaging integrates over parameter values that could have generated the data. Both the number of such parameters with non-zero posterior density (volume) and the degree of support by the data under the given prior (density) govern the Bayesian model average. The *typical set* is the set of parameter states with high posterior mass whose joint density comes close to the total expected density of the parameter distribution. More formally, assume $z_1, \dots, z_N$ are i.i.d. samples from a distribution p on a measurable space. The ϵ -typical set $\mathcal{A}_\epsilon^N$ of the distribution p is given by the set of all length- N sequences whose joint (negative log) density is ϵ -close to the (Shannon) entropy of the distribution:

$$\mathcal{A}_\epsilon^N = \left\{ (z_1, \dots, z_N) : \left| -\frac{1}{N} \log p(z_1, \dots, z_N) - H(p) \right| \leq \epsilon \right\} \quad (33.1)$$

together with

$$H = \mathbb{E}_{Z \sim p} [-\log p(Z)].$$

By the asymptotic equipartition property, the typical set $\mathcal{A}_\epsilon^N$ has probability approaching one as $N \rightarrow \infty$ [337, Ch. 3]. Therefore, sampling it thoroughly is crucial for a faithful assessment of predictive uncertainty: failing to capture the full expanse of models that could have plausibly generated the observed data will leave us overconfident in our predictions. Parameter states outside the typical set, on the other hand, are negligible for inference.

The ease of covering the typical set depends on the structure of the posterior landscape. More complexity in the hypersurface implies a larger number of samples required for accurate inference. Intuitively, the need to visit a large number of modes and collect enough samples from their surrounding area strains the available computational budget. This is especially true when the modes are located far away from each other with areas of low density in between. Such dispersion, as will be discussed shortly, can occur as a consequence of weight-space symmetries, and becomes increasingly likely in higher-dimensional spaces. Samplers will have difficulties crossing the unattractive areas separating modes.

The intractability and high-dimensional nature of neural network posteriors make their analysis, be it visual or mathematical, difficult. We can still approach the problem systematically by recognizing that Bayesian inference bears a connection to empirical risk minimisation when framing the negative log-likelihood (NLL) as a loss function. The widely used *cross-entropy* loss in classification and *L2* loss in regression are such NLL-type objectives. Then, following from Bayes' theorem, the (negative log) posterior landscape is a prior-adjusted version of the *loss*

landscape, up to a normalising constant corresponding to the marginal log-likelihood:

$$-\log \underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} = \underbrace{\hat{\mathcal{L}}(\theta, \mathcal{D})}_{\text{loss}} - \log \underbrace{p_0(\theta)}_{\text{prior}} + \text{const.} \quad (33.2)$$

We will subsequently analyse the respective impact of the loss (or likelihood) and prior component on the posterior landscape.

33.1.2 Impact of Symmetries

Assume for a moment a uniform parameter prior (we will add prior influence to our considerations in an instant). The shape of the resulting posterior density is solely determined by the likelihood in accordance with Eq. (33.2). There are two reasons why the loss landscape can turn out multi-modal, i.e., have more than one local optimum.

1. The observed data support multiple, *functionally diverse* hypotheses. Functional diversity in a given pair of hypotheses $f(\cdot; \theta), f(\cdot; \tilde{\theta}), \theta, \tilde{\theta} \in \Theta$, means that we have $f(x; \theta) \neq f(x; \tilde{\theta})$ for a general input $x \in \mathcal{X}$. Although there is only one true model, the posterior lends support to multiple candidates. This kind of ambiguity is to be expected in finite-data settings where the observed data do not suffice to identify the model that has generated them.
2. Various, *functionally equivalent* parameter states $\theta, \tilde{\theta}$ give rise to the *same* functional mapping, assigning identical network outputs to a given input x , i.e., $f(x; \theta) = f(x; \tilde{\theta})$. This happens precisely in the case of weight-space symmetries as discussed in the previous chapter.

Permutation symmetries are the major driver of symmetry-induced multi-modality. Combinatorially, many equal-loss replicates exist for every functionally diverse mode simply by suitable permutation of weights. Since any two given permutations of a parameter vector can be separated by a large distance in the parameter space, with valleys of low density in between, (naive) MCMC samplers will accrue exponentially long mixing times. Note that such posterior multi-modality occurs not only for our hypothetical uniform prior, but also for the more realistic case of a Gaussian prior with diagonal, isotropic covariance structure. Those priors are widely popular for Bayesian neural networks, usually with a zero mean to encourage parameter sparsity. Multivariate Gaussians with diagonal covariance fully factorise. Together with isotropy, this implies invariance to parameter permutations, as all marginal prior densities are identical. To see this, let $\pi \in \mathbb{R}^{d \times d}$ be a permutation matrix with $\pi^\top \pi = I_d$. Consider a normal random variable $Z \sim \mathcal{N}(\mathbf{0}, \Sigma)$, $\Sigma = \sigma^2 I_d$. Then, the random variable $Z' = \pi Z$ of the permuted inputs retains the original covariance due to the orthogonality of π : $\pi \Sigma \pi^\top = \sigma^2 \pi I_d \pi^\top = \sigma^2 I_d$. The immediate consequence of this phenomenon is the *non-identifiability* of the Bayesian neural network models. From a theoretical perspective, this induces parameter non-identifiability despite a well-defined posterior over functions [904]. Practically speaking, permutation symmetries threaten to make inference vastly more complex and expensive. Since permutation transformations can create symmetric copies in far-flung areas of the parameter space, the resulting modes can be widely dispersed and disconnected.

The prevailing *ReLU* activation $\text{ReLU}(x) = \max(0, x)$ in today's architecture creates a different, continuous type of symmetry (see Chapter 32). It results in level sets of constant-loss parameters that are *connected*. This way, parameter symmetry contributes to the phenomenon of *mode connectivity* [15, 493], describing locally optimal solutions in the weight space linked by paths of near-constant, low loss. Most parameters along such a path produce well-performing models.

ReLU-induced scaling symmetries result in connected areas of constant loss by design. Positive homogeneity of degree one implies the familiar rescaling symmetry, that is, for any hidden layer with ReLU activations, scaling the incoming weights and biases by $c > 0$ and the outgoing weights by c^{-1} leaves the network function invariant. If a hidden unit has pre-activation $a = W^\top x + b$ and outgoing weight vector v , then $(W, b, v) \mapsto (cW, cb, c^{-1}v)$ preserves $f(\cdot; \theta)$ for all inputs. This creates connected level sets of equal loss. A simple two-parameter illustration is the bilinear map $g(\theta_1, \theta_2) = \theta_1 \theta_2$, for which $\{(\theta_1, \theta_2) : \theta_1 \theta_2 = \kappa\}$ is a one-dimensional manifold of equal outputs for any fixed $\kappa \neq 0$, while $\{(\theta_1, \theta_2) : \theta_1 = 0\} \cup \{(\theta_1, \theta_2) : \theta_2 = 0\}$ forms a connected locus of degenerate solutions. Such symmetries can turn isolated optima into extended ridges in parameter space. Beyond permutation and rescaling symmetries, overparameterisation can induce continuous non-identifiability manifolds through redundant neuron splitting; for one-hidden-layer ReLU networks, both the represented function and the quadratic Gaussian-prior penalty remain unchanged along these manifolds, so that parameter uncertainty can persist even when the underlying function is fully identified [908, 1439].

Empirical evidence suggests that, for sufficiently large models, there is a sizeable manifold of good solutions that is connected save for permutation symmetries [15, 1517]. We thus have two forces potentially acting on

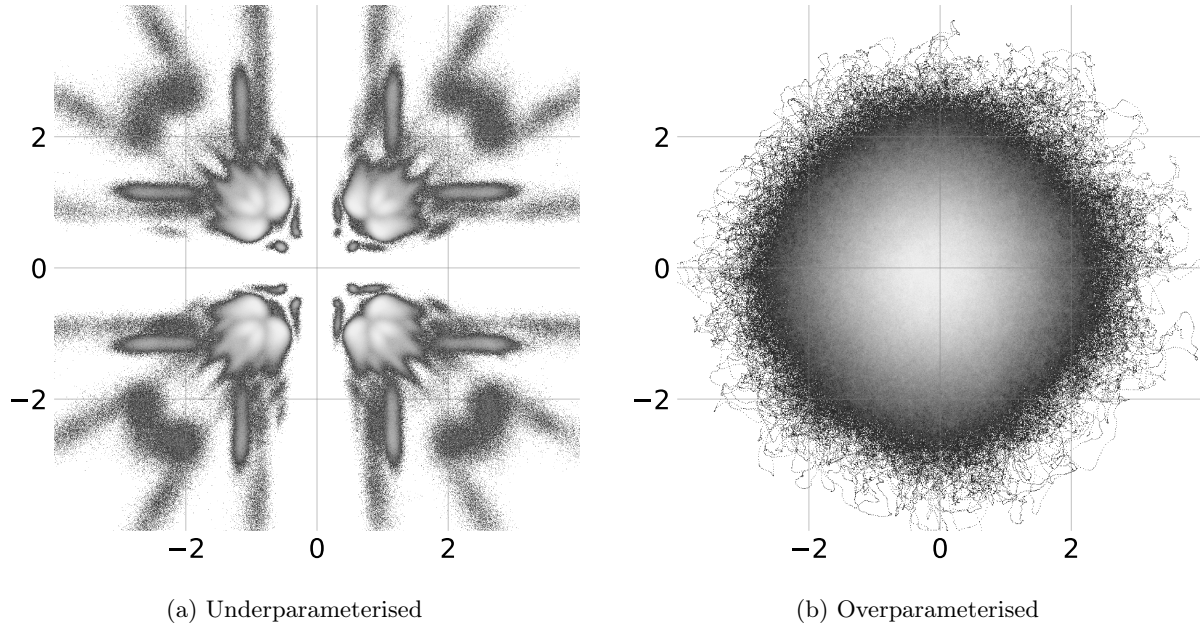


Figure 33.1: Eight million posterior samples for two different networks. The plots show the bivariate marginal weight distribution of two randomly picked weights from the same hidden layer. The left panel corresponds to a single hidden layer network with three neurons. The right panel corresponds to a three hidden layer network with sixteen units per layer. In the underparameterised setting, a clear symmetric structure is visible, which poses a challenging sampling problem. In the overparameterised setting, the strong influence of the chosen standard Gaussian prior yields almost perfect marginal prior conformity. The figure is adapted from [908] and sampling is performed using MILE [1535]

the degree of multi-modality in the loss landscape: permutation symmetries creating disconnected local optima, and rescaling symmetries leading to connected constant-loss volumes. In contrast to permutation symmetries, non-uniform priors that favour certain parameters over others may break *rescaling* symmetries in the posterior landscape. Zero-mean Gaussians in particular prefer parameter vectors with small norm and assign progressively less weight to parameters of larger magnitude.

33.1.3 Prior Influence

Eq. (33.2) shows that the posterior density is controlled by a combination of the loss (or likelihood) and the prior. Symmetries, depending on their type and extent, can introduce multi-modality in the loss landscape. The commonly used prior densities (especially of the Gaussian sort), on the other hand, are convex and uni-modal. The relative influence of prior and likelihood depends on the degree of *overparameterisation* and on the informativeness of the data. Increasing the number of parameters for a fixed dataset often increases the relative influence of the prior, because the likelihood becomes more diffuse across directions unconstrained by the data [904, 908, 1439].

Fig. 33.1 illustrates the effect of prior influence. The left scenario belongs to the underparameterised regime and displays strong symmetric patterns in the posterior density. The overparameterised case on the right shows an entirely different picture, with the marginal posterior density closely matching the chosen isotropic standard Gaussian prior. It is easy to imagine that the left-hand scenario poses more of a challenge to sampling-based inference. In this situation, the posterior hypersurface is much less convex and connected, yet highly structured by visible symmetries.

The interplay between the prior and the data in the light of symmetries and overparameterisation affects the posterior in non-trivial ways. Paradoxically, *larger* models might be more amenable to sampling-based inference because the prior—at least, for the popular Gaussian variant or similarly smooth alternatives—has a softening effect on the posterior landscape [908]. Overparameterisation seems to be similarly beneficial to the curvature by promoting flatter optima [1328]. In practice, prior choice and model selection remain the key design challenges of the modeling process and need to be evaluated carefully with measures of calibration and predictive performance.

33.2 Sampling Strategies

How can the insights about the posterior landscape be used to devise good sampling strategies? The functional equivalence of symmetric parameter states begs the question of whether samplers actually need to visit *all* the modes scattered through the parameter space. Early works in the 1990s that first pointed out the existence of functionally equivalent parameter states in the loss landscape implied the existence of *minimally sufficient search sets* [265, 724]. These sets contain representatives from each functionally diverse parameterisation, of which all other states are mere symmetric replicates.

In theory, confining inference to the sufficient sets and re-scaling posterior dispersion appropriately should serve to cover all functionally relevant parameters. For discrete symmetries (in particular, those caused by parameter permutations), this can be translated to the following, modified version of the posterior predictive density [1748]:

$$\begin{aligned} p(y_*|x_*, \mathcal{D}) &= \int_{\Theta} p(y_*|x_*, \theta) p(\theta|\mathcal{D}) d\theta \\ &\approx |\mathbb{T}| \int_{\mathcal{S}} p(y_*|x_*, \theta) p(\theta|\mathcal{D}) d\theta, \end{aligned} \quad (33.3)$$

assuming a finite $\mathbb{T}$ -invariant posterior with $\mathbb{T}$ the set of output-preserving transformations,

$$\mathbb{T} = \{\mathcal{T} : \Theta \rightarrow \Theta \mid f(x; \theta) = f(x; \mathcal{T}(\theta)) \quad \forall x \in \mathcal{X}\}, \quad (33.4)$$

and a disjoint tiling of Θ by the orbits (see Def. 32.7 in the previous chapter) of $\mathbb{T}$ up to null sets. It should be noted that, for general symmetries, the transformations $\mathbb{T}_\theta$ that preserve the network output depend on θ . Lastly, $\mathcal{S}$ denotes the minimally sufficient search set with $|\mathcal{S}| \ll |\Theta|$. Defining $\mathcal{S}$ explicitly can be difficult in practice. More importantly, there is no straightforward method to restrict sampling to this minimally sufficient search set. That said, several approaches exist to sample some—small and accessible—subspace of the entire parameter space (see 33.2.2).

33.2.1 Symmetry Removal

Eq. (33.3) and the concept of minimally-sufficient search sets suggest that the *removal* of symmetries might facilitate the inference problem on a great scale—with the number of permutations increasing exponentially in higher dimensions, the amount of symmetric copies usually exceeds the number of functionally diverse solutions by far. Similar considerations follow from the insights on mode connectivity, suggesting that breaking permutation symmetries will leave us with a connected, easy-to-traverse hypersurface. Several ideas exist to remove symmetries [1328, 1564, 1748, see also Chapter 32]. As symmetries are vastly complex and hard to characterise for arbitrary architectures, especially when overlapping in their effects, it is challenging to design general-purpose algorithms to remove them fully.

33.2.2 Subspace Inference

A parameter space devoid of symmetries constitutes a *subspace* of the original. While recovering the symmetry-free subspace is hard, other ideas exist to find subspaces amenable to sampling (typically of lower dimension; see also Chapters 39, 48). Subspace inference can replace the full procedure if the subspace is chosen in a meaningful way and all functional diversity is captured. This can be achieved, for example, by finding the most important parameters as indicated by the variability along the respective dimension in a loss-based optimisation procedure [1116]. As an alternative, we can use low-loss tunnels resulting from mode connectivity approaches, which are typically low-dimensional by design [436, 825].

33.2.3 Multi-Start Sampling

Other approaches propose to sample the original weight space but in a clever way. Most practical problems have posteriors with multiple basins of attraction. A simple yet effective approach is then to run several samplers in parallel [1534, 1535, 1765]. Even when choosing the starting locations for the chains at random, this can potentially improve performance as a single-chain sampler can get stuck or start exploring irrelevant areas of the parameter space. Such a multi-start sampling approach can be further improved by smartly choosing the initial chain locations. Successful approaches use, for instance, deep ensembles to warm-start sampling-based inference. For a detailed discussion of such hybrid inference methods, see Chapter 20. Other than choosing favourable starting locations, we can incentivise samplers to explore the search space more effectively by enforcing restarts during the

sampling stage. *Cyclical MCMC* [1854, Chapter 2] does so by scheduling the learning rate to alternate between exploration (large moves to cross the space in search of different modes) and exploitation (small steps to cover the local area around a single mode). A similar idea exists in non-Bayesian deep learning via so-called *symmetry teleportation*, allowing the optimiser to jump towards solutions with identical loss but steeper gradient by means of output-preserving parameter transformations [1864].

Generally speaking, the impact of parameter symmetries on sampling-based inference varies across different practical situations—underparameterised problems may exhibit pronounced symmetric patterns, while overparameterised models lean towards smoother, more prior-dominated posteriors. The need and benefit of accounting for symmetries in the inference procedure thus depends strongly on the problem at hand.

33.3 Sampling diagnostics in the presence of symmetries

Assessing the convergence of MCMC samplers is a cornerstone of the practical Bayesian workflow and has received renewed interest in the light of modern sampling applications [964, 1131, 1195]. We aim to run the sampler long enough for it to reach its stationary distribution, so that the resulting samples can be used to approximate the posterior reliably. However, the presence of weight-space symmetries in Bayesian neural networks violates the assumptions underpinning classical MCMC diagnostics, which can lead to misleading conclusions about convergence. This section examines why these standard measures often fail and discusses the challenges of diagnosing convergence in non-identifiable models.

33.3.1 Diagnostics in Parameter Space

The two most common diagnostics for MCMC convergence are the potential scale reduction factor, $\hat{R}$, and the effective sample size (ESS), both of which have been introduced in Chapter 1. However, $\hat{R}$ and ESS have been designed for models with identifiable parameters, a condition that does not hold for BNNs.

The Potential Scale Reduction Factor ($\hat{R}$)

The $\hat{R}$ diagnostic is designed to determine whether multiple, parallel MCMC chains have converged to the same distribution. To this end, the diagnostic compares the variance of samples *between* the chains and the variance *within* each chain. More formally, for K chains of length S , the statistic is based on the between-chain variance (B) and the within-chain variance (W):

$$B = \frac{S}{K-1} \sum_{k=1}^K \left(\bar{\theta}^{(k,\cdot)} - \bar{\theta}^{(\cdot,\cdot)} \right)^2, \quad W = \frac{1}{K} \sum_{k=1}^K \mathbb{V}(\theta^{(k,\cdot)}).$$

$\bar{\theta}^{(k,\cdot)}$ denotes the sample mean of the k -th chain, and $\bar{\theta}^{(\cdot,\cdot)}$ is the overall mean of samples across chains. The potential scale reduction factor is then given by:

$$\hat{R} = \sqrt{\frac{\hat{\mathbb{V}}^+(\theta|\mathcal{D})}{W}} := \sqrt{\frac{\frac{S-1}{S}W + \frac{1}{S}B}{W}}. \quad (33.5)$$

Here, $\hat{\mathbb{V}}^+(\theta|\mathcal{D})$ is an estimate of the marginal posterior variance. The '+' in $\hat{\mathbb{V}}^+$ signifies that it is an intentionally overestimated (and thus more conservative) estimate, created by combining the within-chain (W) and between-chain (B) variances. If the chains have not fully explored the posterior, W may be an underestimate of the true variance, and this combined measure helps correct for that.

For greater robustness, modern implementations compute this statistic on modified samples [1672]. First, the calculation is performed on folded chains, meaning each chain is split into subchains. This effectively increases the number of chains considered in the calculation and allows the diagnostic to better detect non-stationarity (i.e., if a chain's first and second halves explore different regions). Second, the statistic is computed on rank-normalised samples to further increase robustness.

A value of $\hat{R}$ close to 1.0 (e.g., Vehtari et al. [1672] suggest a threshold of < 1.01) is taken to indicate convergence, as it implies that the between-chain variance (B) is negligible compared to the within-chain variance (W).

The rationale of $\hat{R}$ breaks down in the presence of symmetries. Consider that we are using two chains to sample from a BNN posterior. Due to different random initialisations, they will likely explore different but potentially

functionally equivalent modes. For instance, one chain might explore a mode centred around a parameter θ_A , while the second explores a symmetric mode centred at θ_B . Even if both chains have perfectly converged locally and are stationary within their respective modes (thus also functionally), the parameter means $\bar{\theta}_A$ and $\bar{\theta}_B$ will be different. This results in a large and persistent between-chain variance B , causing $\hat{R}$ to remain bounded away from one even after very long runs. The diagnostic will therefore falsely signal non-convergence, when in reality, the chains have successfully identified the local posterior geometry. Consequently, applying the standard multi-chain $\hat{R}$ to the parameters of a BNN is inappropriate.

Effective Sample Size (ESS)

The effective sample size (ESS) estimates the number of independent samples that are information-theoretically equivalent to the autocorrelated samples produced by an MCMC chain. With $K \cdot S$ total samples, possibly gathered from K parallel-running chains of length S , it can be defined as:

$$\text{ESS} = \frac{KS}{1 + 2 \sum_{t=1}^{\infty} \rho_t}, \quad (33.6)$$

where ρ_t is the autocorrelation of the samples at lag t . Most practical implementations use Geyer corrections to stabilise the ESS computation, as the estimate can become noisy for high t (only few samples are separated by such large lags and can contribute to the autocorrelation estimation). In sampling, high autocorrelation is undesirable and leads to a low ESS, suggesting inefficient exploration.

In BNNs, symmetries create continuous manifolds, or “ridges”, in the posterior landscape where the probability density is constant. A sampler moving along such a ridge may exhibit random-walk behaviour, which induces high autocorrelation in the parameter traces. This can result in very low ESS values even if the sampler is efficiently traversing the functionally important parts of the space. Therefore, like $\hat{R}$, a low ESS can have various causes: it may indicate that the sampler has not yet converged, or a genuinely “stuck” sampler unable to escape from a local basin, or simply reflect the sampler’s movement along a symmetric manifold.

33.3.2 Diagnostics in Function Space

Given the aforementioned issues in parameter space, one might hope that assessing convergence in function space—by analysing the model’s outputs—would be more meaningful. The general idea is to apply diagnostics not to the parameters θ , but to some function of them, such as the predictions on a held-out dataset or a performance metric like the log-likelihood. However, this approach faces a similar, albeit more subtle, conceptual problem. While symmetries create functionally identical modes, BNN posteriors are also known to contain *functionally distinct* modes [1748]. It is common for different MCMC chains to stabilise at different but locally consistent performance levels [1534]. While these solutions all have high posterior probability, they produce functionally different, high-performance models. If we compute a functional $\hat{R}$ on the predictions of these chains, we again penalise between-chain disagreement. A high $\hat{R}$ value might simply indicate that the chains have successfully discovered different, competing functional modes. From a Bayesian perspective, capturing this functional diversity is often a primary goal of the analysis, as it reflects genuine model uncertainty. A diagnostic that punishes the discovery of good but different solutions therefore creates the wrong incentive; it mistakes successful exploration of a complex function space as a failure and rewards sampling redundant modes. This issue persists for both pointwise predictions and summary statistics like the root mean squared error or the log pointwise likelihood.

33.3.3 Towards More Meaningful Diagnostics

The failure of classical diagnostics necessitates a shift in perspective. Rather than asking “Have all chains converged to the same answer?”, a more appropriate set of questions for BNNs would be:

1. Has each individual chain reached a stationary state?
2. How diverse is the ensemble of solutions found by the different chains?

To address the first question, diagnostics can be adapted to operate on a **chain-wise** basis. For example, one can split a single chain into multiple segments and compute a “chain-wise” $\hat{cR}$ to check for internal stationarity as proposed in Sommer et al. [1534] and closely connected to the rationale presented by Vehtari et al. [1672]. For the second question, one can monitor the evolution of a chain-wise performance metric, such as an expanding-window estimate of the log posterior predictive density. This allows checking whether each chain has stabilised in terms of its predictive performance and quantifying the disagreement between the final performance levels of the chains.

It is crucial to recognise, however, that these adapted methods are not fail-safe. A chain can appear stationary while being stuck in a small, suboptimal region of a posterior mode. At present, developing robust, scalable, and theoretically sound diagnostics for non-identifiable models remains an open and critical research challenge in the field of Bayesian deep learning.

Chapter 34

Learning inductive bias and symmetries

Backpropagation is the cornerstone of current deep learning practice. If you specify a neural network architecture and a dataset to train on, backpropagation is used to find weights that make good predictions. However, designing an architecture is currently still a laborious process, which relies on manual design, trial-and-error, and crossvalidation. In this chapter, we investigate how Bayesian inference can help automate this procedure, with the goal of having adaptable architectures that can be learned as easily as the weights.

Having such a training procedure would give numerous advantages. The most straightforward is that, when faced with a new dataset, training would need less human design and supervision. A fully automatic method would mean that practitioners would no longer need to worry about selecting the hyperparameters that control the architecture, such as the number of layers, widths of layers, pooling structures etc. Indeed, making learning automatic is one of the main points of machine learning in the first place! Such procedures could also be more computationally efficient, as current model adaptation requires many iterations of training from scratch. There could be significant computational savings if these could be combined in a single training run, even if this one run is more expensive than a single run on a fixed architecture.¹

We formulate our goal in terms of **three desiderata**:

1. We want our procedure to work well over as wide a range of tasks and settings as possible (i.e. reliably), and with as little built-in prior knowledge and human input as possible (i.e. autonomously).
2. We want learning of hyperparameters (e.g. controlling architecture) to be as easy as learning the weights. This means that only a single training dataset should be needed (rather than train/validation splits), and that a single procedure should determine both the weights and hyperparameters (e.g. unified training objective).
3. We want the generalisation performance of our resulting model to be as good as possible. Crossvalidation is a good baseline that we do not want to sacrifice too much performance from.

Currently, there is no method that perfectly fulfils all these desiderata. However, these desiderata do help assess methods, and guide research towards improvements.

A learning procedure for neural network architecture, like all learning procedures, will need to consist of two ingredients: **(i)** a space of candidate solutions, and **(ii)** a procedure for selecting within this space. To make ingredient **(i)** manageable in practice, we will take inspiration from Geometric Deep Learning [210, 318, 914, 1735], which studies how to create architectures with vastly different inductive biases by enforcing symmetries in the functions that neural networks encode. Based on these ideas, we will parameterise architectures that interpolate between different symmetries. This allows us to build architectures that have the ability to move between different architectures during training, where they would normally need be manually designed and fixed.² For ingredient **(ii)**, we will show how this is a *model selection* problem that Bayesian hierarchical inference can help with. Bayesian reasoning directly provides a pathway to selecting a good architecture, that is much more robust than the normal training loss, and works towards satisfying our desiderata more than crossvalidation.

Put together, we obtain a single architecture that can automatically adapt its inductive bias to the dataset that it is trained on. We show that if the *same* adaptive method is run on datasets with different characteristics,

¹A reasonable question within the current paradigm of scaling laws, is whether such tuning is necessary. The strategy of picking a single architecture and scaling it is currently leading to continuous improvements. At the end of this chapter we will acknowledge these trends, and motivate future research directions.

²Our space of architectures does not include all symmetries or architectures, so the lofty goal from recognising symmetries or building architectures completely from scratch is still open. Parameterising a space of architectures that is wide enough to include symmetries, but is not too big to make learning infeasible is still an open research problem, as we will discuss later.

the architecture will adapt its inductive bias to respect the symmetry that is present in the dataset. Crucially, it does this using only a training dataset, without any nested training procedures, in a single run of training, and so making progress on obtaining a method that satisfies our three desiderata.

In the course of this chapter, we will:

- formalise our problem and review why the regular training loss is not used to learn architecture (Section 34.1),
- discuss other approaches to adapting inductive bias in DNNs, paying special attention to Neural Architecture Search, and assess to what degree these frameworks can satisfy our goals,
- review Bayesian model selection, how it has been proposed in the past to learn inductive bias, and its possibilities if it could be applied to neural networks,
- explain a line of work that uses (approximate) Bayesian model selection to automatically adapt neural network architecture, and finally
- discuss the limitations of Bayesian model selection, and the relevance of adaptive architectures to the current trend of large, scalable models.

Throughout this chapter, one question will recur: Does the Bayesian approach offer benefits beyond what can be achieved with point estimates? This question is important, since developing Bayesian methods adds computational cost and complication to a deep learning system³, and the research endeavour is only worth it if sufficient benefits can be obtained. Whether this is possible in practical deep learning settings is an open research question.

This chapter covers the current evidence for answering this question from two perspectives: **1)** what Bayesian inference promises to deliver *if the requisite quantities can be computed*, and **2)** what is achieved by currently implementable methods. The first perspective has long been discussed in work on Bayesian model selection, hierarchical Bayesian inference, and Occam’s razor. We discuss the ongoing debate, and provide our view is that the framework that was set up in the past is mostly correct, and that the greatest challenges lie in actually calculating the necessary quantities. To provide evidence from the second perspective, we discuss a line of work [812, 815, 1658, 1660–1662] that approximates what Bayesian model selection prescribes to do, and compare it to alternative approaches.

We, the authors of this chapter, are co-authors of this line of work. We believe that the results are encouraging: Bayesian thinking can lead to new capabilities in deep learning, and the closer we approximate what Bayes prescribes, the better our methods work. However, many more steps are needed to make these methods ready to be routinely applied in deep learning practice. In this chapter, we summarise the existing evidence and, while acknowledging that the research question remains open, argue that this direction warrants further exploration, with the hope of inspiring others to join the investigation.

34.1 The Problem Setting

To begin, let’s review why architecture selection is currently performed. Before investigating any methods, it is helpful to first review how the problem is currently approached, to highlight challenges that any potential method will need to deal with.

Regardless of whether we are solving toy 1D regression examples, classifying MNIST or ImageNet, or predicting the next token in an LLM, neural networks solve the same problem: function estimation. From a dataset of input-output pairs $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ (with $x_n \in \mathcal{X}, y_n \in \mathcal{Y}$), we aim to find a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that makes good predictions at new inputs.⁴ “Good” is typically defined by achieving a low loss on future data, where future data can arrive i.i.d. from the same distribution as the training data $\mathcal{D}$, or from some kind of new process, as is the case in the distribution shift or out-of-distribution (OoD) settings.

Deep learning works by first selecting a neural network architecture, which is controlled by various hyperparameters, for example, number of layers, type of layer (e.g. fully connected vs convolutional), connectivity pattern (e.g. stride or window size) width, regularisation, skip connections, etc. The architecture can be seen as a map from the weights to a function from inputs to outputs which we can denote as

$$f^\theta : \mathbb{R}^{D(\theta)} \times \mathcal{X} \rightarrow \mathcal{Y}, \quad (34.1)$$

$$f_w^\theta : \mathcal{X} \rightarrow \mathcal{Y}, \quad (34.2)$$

³In particular, we will see that hierarchical Bayesian inference requires approximating integrals that quantify *volume* of the parameters that fit the training set, rather than just keeping track of a single parameter that fits the training set.

⁴Typically $\mathcal{X} = \mathbb{R}^D$, and for discrete classes/tokens as outputs we have $\mathcal{Y} = \{0, 1\}^C$, while for our visualisations we take $\mathcal{Y} = \mathbb{R}$.

with θ controlling the architecture, and $w \in \mathbb{R}^{D(\theta)}$ being the weights (with dimension potentially depending on the architecture). After specifying an architecture, the weights are chosen by (approximately) minimising some loss w.r.t. the parameters

$$w_\theta^* \approx \arg \min_w \sum_{n=1}^N \ell(f_w^\theta(x_n), y_n) \quad x_n, y_n \in \mathcal{D}, \quad (34.3)$$

which is usually performed by running some kind of (stochastic) gradient descent for certain number of iterations. We write w_θ^* to make the dependence of the optimal weights on the hyperparameters θ explicit.

Given that the weights are optimised on the training data, sufficiently flexible architectures can often achieve similar training losses,

$$\sum_{n=1}^N \ell(f_{w_\theta^*}^\theta(x_n), y_n) \approx \sum_{n=1}^N \ell(f_{w_{\theta'}^*}^{\theta'}(x_n), y_n). \quad (34.4)$$

Particularly for flexible architectures and low-noise datasets, training losses can be near zero, making this very close to equality. However, there are many different functions that go near the training data, but which take on different values for other inputs, as shown in Fig 34.1. *Any* prediction method must choose which of the many functions that minimise the loss to pick. Which function is chosen is determined by the *inductive bias*.⁵ In neural networks, the architecture influences the inductive bias, given that it determines how weights map to an entire function (Eq 34.1).

This leads to the question: How should inductive bias (and therefore architecture) be selected? This is the central question of this chapter, and our goal is to find a procedure that makes this selection as easy as selecting the weights.

34.2 Potential Methods

34.2.1 Minimising the Training Loss: Flexible Models Fit Any Data, But Generalise Differently

To start, we can ask why minimising the training loss, which works so well for the weights, is not also used to learn hyperparameters. One could easily perform the minimisation

$$\theta^*, w^* \approx \arg \min_{\theta, w} \sum_{n=1}^N \ell(f_w^\theta(x_n), y_n) + n_\theta(w) \quad x_n, y_n \in \mathcal{D}, \quad (34.5)$$

through some (stochastic) gradient descent method, perhaps relying on some additional regularisation term $n_\theta(w)$. If this were possible, adjusting architecture *would* be as easy as learning the weights!

The reasons for why this method does not work in general are often discussed in the earliest stages of an ML course [160, p. 11], and show why validation sets are needed. Nevertheless, in specific cases it has sometimes been used to learn hyperparameters [e.g. 120, 1418, 1657], and there are different mechanisms that cause it to succeed or fail (see Section 34.2.3).

The core reason for not using the training loss to learn hyperparameters is that, inductive biases that help generalisation do not necessarily improve the training loss. The regularisation term $n_\theta(w)$ is an additional loss, which can be removed by reducing the hyperparameter controlling its weight. In addition, architectures with different inductive biases are still flexible enough to fit the training data, so the training loss cannot distinguish them even though they generalise differently.

An example of this is shown in Fig 34.1, which compares the train and test loss of two models with and without a periodic inductive bias, on a dataset with periodicity. Both models were chosen to be overparameterised, in accordance with the regime that deep learning operates in.⁶ In addition, the training set was taken to be noise-free,

⁵Jumping ahead a bit, it may seem that Bayesian machine learning avoids choosing a function to pick, by instead selecting a distribution over possible functions. It would be a mistake to conclude that Bayesian methods are therefore less reliant on inductive biases. There are still many distributions over functions that fit the training data, which all predict differently, and the inductive bias determines which of these distributions a Bayesian method picks.

⁶The idea that models should have more parameters than datapoints was considered puzzling in deep learning. Belkin et al. [110] coined the term “double descent”, although this behaviour was also noted and explained earlier (e.g. Bős and Oppel [189]). Using more parameters than datapoints is no surprise in Bayesian nonparametrics though, e.g. in Gaussian processes where the number of weights is infinite [1380, 1382].

and hyperparameters controlling e.g. the smoothness were set to values that were optimal for generalisation. We see that the periodic model performs much better on the test set, but has an effectively identical training loss as the non-periodic model, which performs much worse on the test set. This demonstrates that the training loss does not correctly distinguish the right model in this situation.

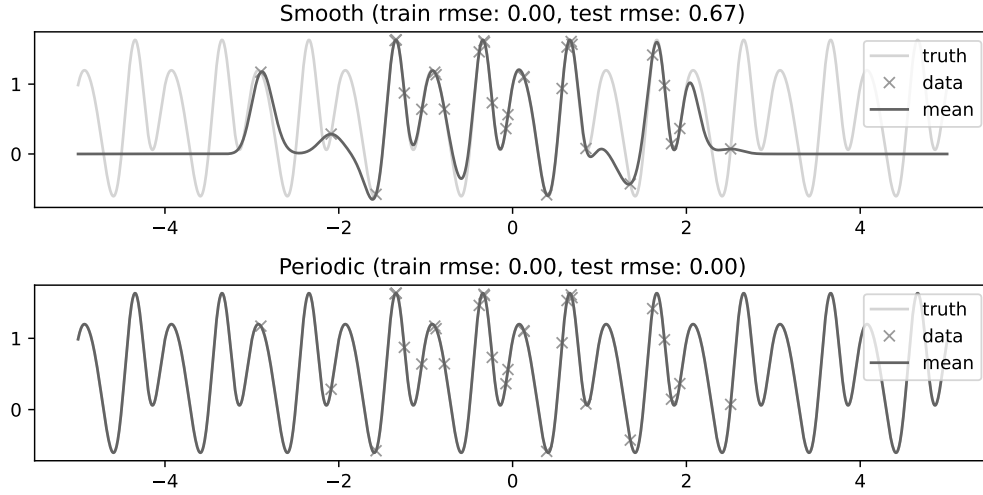


Figure 34.1: Learning a function that has a periodicity in it, from noise-free data. We show two models with different inductive biases: smoothness (top), and smoothness *plus* periodicity (bottom). Both models are overparameterised and cannot be distinguished according to their fit on the training data, but test loss is much lower for the model with the appropriate inductive bias. Both models are single-layer finite basis-function models, with weights minimising the training loss without regularisation, and the hyperparameters (e.g. lengthscale of the smoothness) set to values that lead to optimal test loss. While the training losses are best described as indistinguishable, it is worth noting that the loss for the smooth model ($8.6 \cdot 10^{-6}$) is smaller than that of the periodic model ($2.8 \cdot 10^{-5}$).

This effect is made worse the larger the variation in train-test loss gap between models. Uncontrolled overfitting causes this, and the experimental setting was chosen to minimise this overfitting by selecting good hyperparameters for each model, and using a low-noise dataset. Noise only increases this effect, since noise is very unlikely to fit with inductive biases, leading to the best-fitting model being the most flexible one.

34.2.2 Crossvalidation: The Common Solution

From the first classes in ML, crossvalidation is proposed as a solution to the problems with the training loss. The method is justified by selecting the inductive bias using a direct estimate of the performance on future data, following our original goal. We estimate this “validation loss” for several hyperparameters θ on a “validation” dataset that was unseen during training, and select the inductive bias with the best estimated performance:

$$\theta^* = \arg \min_{\theta} \sum_{n=1}^{N_{\text{test}}} \ell(f_{w_{\theta}^*}^{\theta}(x_n), y_n), \quad (34.6)$$

noting that w_{θ}^* is still selected by minimising the training loss according to Eq. 34.3. We perform this minimisation by evaluating the validation loss for several different hyperparameter values, and then selecting the best one. Since the weights depend on the hyperparameter, this requires retraining the weights separately for each hyperparameter value that is considered. This whole procedure is known as (1-fold) crossvalidation, and is effectively trial-and-error.

Provided that the validation set size is large enough to fit the desired number of hyperparameters⁷, crossvalidation is highly reliable and near-ubiquitously applied in machine learning. The machine learning community as a whole can also be seen as a large crossvalidation process, with researchers across the world suggesting new architectures, and publishing those that perform better on a test set.

⁷If a small validation set is used to select a large number of hyperparameters, crossvalidation can also end up overfitting [248, 1234]. Other variants of crossvalidation (e.g. K-fold even leave-one-out) can reduce this overfitting [830, §5.1], at the cost of needing to re-train the w_{θ}^* multiple times. However, the number of hyperparameters is typically small.

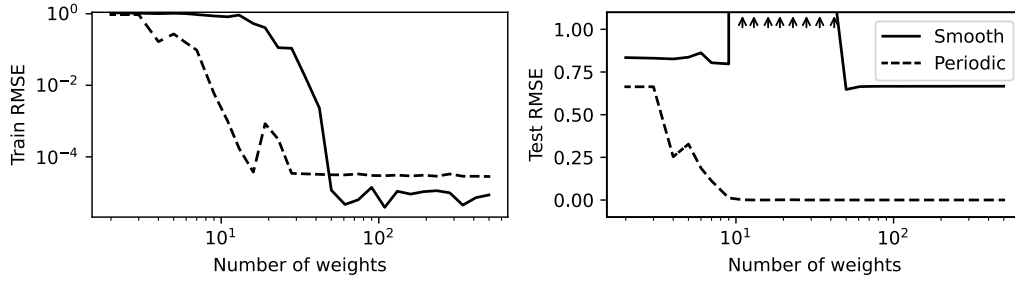


Figure 34.2: Training and test losses for smooth and periodic basis function models as the number of weights increases. With limited capacity, the periodic inductive bias improves both training and test loss. Once both models are sufficiently overparameterized, their training losses become indistinguishable while their test losses remain different, so training loss no longer identifies the better inductive bias.

Despite the ubiquity and success of this process, it requires humans to design new architectures and use a multi-level training procedure, which breaks desiderata (1) and (2). Nevertheless, the reliability and performance of crossvalidation’s results make it the current gold standard. Any method for selecting inductive bias cannot sacrifice too much performance, or it will not be used, as described in desideratum (3).

34.2.3 When Training Losses *do* Identify Inductive Bias

The desire to avoid the inconvenience and computational cost of using validation sets has led to attempts to still use the training loss to provide signal for learning hyperparameters (e.g. Romero and Lohit [1418]). Some success has been reported, in the sense that training over some additional hyperparameters leads to performance gains over fixed hyperparameters. These results are certainly better than what is implied by Fig 34.1, where selecting hyperparameters in this way would lead to an actively worse model. Given these results, we must ask: In what situations do training losses provide useful signal for learning hyperparameters?

One mechanism that can cause this method to work, is implicit regularisation of the solution by the initialisation and the optimisation dynamics. Consider a model that can select between the smooth and periodic models of Fig. 34.1:

$$f_w^\theta(x) = e^{\theta_1} f_w^{\text{smooth}}(x) + e^{\theta_2} f_w^{\text{periodic}}(x) \quad (34.7)$$

$$\theta^*, w^* = \arg \min_{\theta, w} \sum_{n=1}^N \ell(f_w^\theta(x_n), y_n) - c_1 e^{\theta_2} + c_2 e^{\theta_1}. \quad (34.8)$$

If the model is initialised with the most restrictive inductive bias (e.g. a large θ_2 and small θ_1 in the equation above), then there will be encouragement to move to a more flexible model if the inductive bias is inappropriate (and significantly hurts training loss), while the inductive bias may remain largely unchanged if it is appropriate (and still allows the data to be fit), particularly if there is some early stopping.

While this approach *can* work, it is brittle, and relies on a-priori input or human tuning to determine what initialisation and learning rate to use, or what the right amount of early stopping is. Both of these may also require some trial-and-error (or crossvalidation) to get right.

Another mechanism is the interaction between model size and inductive bias. There are settings where an appropriate inductive bias actually helps a small model to fit the data. In these cases, if the model size is constrained, then the training loss will reliably prefer the model that generalises well.⁸ We ran the example from Fig 34.1 for different model sizes, and plotted the train and test losses in Fig 34.2. We see that when the number of weights is constrained, the training loss is lowest with the periodic inductive bias, leading to it preferring the model that generalises well. On the one hand, this can be seen as a good thing: Surely we want small models that generalise well?

However, for this approach to be reliable, it requires *an appropriately small model size to be set a priori*. The size must be large enough to not limit performance of the model with the good inductive bias, but small enough to prevent the training loss from preferring the more flexible model with the worse inductive bias. This “Goldilocks” is difficult to know before training, and therefore may again need crossvalidation to determine. Moreover, models

⁸The effect of model size influencing inductive bias has also been noted in transformers [1070].

are typically trained in the overparameterised regime, meaning that this mechanism for the training loss to work is brittle in exactly the regime where it matters.

The reliance of both these mechanism on a-priori knowledge or tuning means we do not consider them robust enough to satisfy desiderata (1) and (2). However, the existence of these mechanisms does provide a difficulty for evaluating such methods. If we see a method based on training loss working, we should carefully assess whether

1. we are perhaps using an intialisation based on information that is not available when the method is applied in practice, and
2. the model size is constrained and too small for allowing the architecture to predict optimally,

before concluding a method is generally robust. Although, the converse warning also holds, in that many heuristics have been remarkably successful in spite of earlier theoretical objections.

34.2.4 Explicit Regularisation of Hyperparameters

So far, we have noted various ways in which the training loss of Eq 34.3 can lose or give poor training signal to select inductive biases, in models that are sufficiently flexible to fit the training data. A natural idea to combat this is to add an *explicit regularisation* term $r(\theta)$ to directly encourage the selection of models that are known to generalise better:⁹

$$\theta^*, w^* \approx \arg \min_{\theta, w} \sum_{n=1}^N \ell(f_w^\theta(x_n), y_n) + n_\theta(w) + c \cdot r(\theta) \quad x_n, y_n \in \mathcal{D}, \quad (34.9)$$

For the example in Fig 34.1, we could have created a model that mixed smooth and periodic components, and used a training loss that encouraged periodicity:

$$f_w^\theta(x) = e^{\theta_1} f_w^{\text{smooth}}(x) + e^{\theta_2} f_w^{\text{periodic}}(x) \quad (34.10)$$

$$\theta^*, w^* = \arg \min_{\theta, w} \sum_{n=1}^N \ell(f_w^\theta(x_n), y_n) - c_1 e^{\theta_2} + c_2 e^{\theta_1}. \quad (34.11)$$

This explicitly encourages the function to include a large periodic component. If this periodic component does not hurt the fit to the data too much, it is preferred. Many symmetry discovery methods use an objective function of this form, including Augerino [120] for learning affine invariances, Residual Pathway Priors [521], learning local symmetries in [149], symbolic differential equations [1799], explicitly learned weight-sharing in [1657], and bispectral networks [1457].

This strategy does come with several drawbacks:

1. A-priori knowledge is needed of which inductive biases will work on the task at hand, so the correct encouragement and discouragement can be encoded in $r(\theta)$.
2. The regularisation depends on the parameterisation of the model, and changes to the parameterisation can completely change a regulariser's behaviour (e.g. $\theta' = 1/\theta$). As a consequence, regularisers must be tailored to a particular model.
3. Additional hyperparameters (e.g. such as the strength of the regularisation c above) are introduced, which may require tuning in itself.

The a-priori knowledge that is required makes this approach task dependent, and reliant on an external/human source of this knowledge, breaking desideratum (1). In addition, any need for additional tuning of hyperparameters breaks desideratum (2).

The *degree* to which these drawbacks are a problem in practice depends on the sensitivity to the above factors, which can be determined empirically. If a single regularisation gives improved results over a wide range of tasks, it will still be useful. Some evidence of success exists in the literature: When the loss landscape contains broad, near-flat basins of almost constant training loss the sensitivity to the tuning of the hyperparameter regulariser $c \cdot r(\theta)$ may be low [119, 521, 1659]. However, empirical examples of the failure modes also exist (see Sec. 34.8.5, or App. C in [815]). Ultimately, the only way to determine the sufficiency of a method in practice, is an experimental comparison.

⁹Probabilistically speaking, this can be interpreted as placing a hyperprior $p(\theta) \propto \exp[-c \cdot r(\theta)]$ on the inductive-bias parameters, when the loss is properly scaled and the resulting density is normalisable.

34.2.5 Differentiable Validation Losses and Neural Architecture Search

Overall, of the methods discussed so far, crossvalidation remains the most widely applied, at the cost of requiring retraining models many times. Another line of inquiry focusses on reducing this cost, by estimating gradients of the validation loss w.r.t. hyperparameters. Notable examples are in the field of neural architecture search (NAS). Notably, Darts [1060] does this by differentiating the validation loss through a finite sequence of updates on weights based on training data. This requires backpropagation through the optimization trajectory, which involves Hessian-vector products. Alternatively, this can be done by implicit differentiation at a converged inner optimum involves an inverse-Hessian-vector product [1080]. Both approaches require approximations to allow application in practice. Methods based on neural architecture search also often use evolutionary algorithms [490, 1384] or reinforcement learning [1885]. Applied to symmetry discovery, notably [1869] differentiate validation loss to learn convolutional weight-sharing structure and AutoAugment [348] used reinforcement learning outer loops. Other approaches to symmetry discovery also exist, for example through generative adversarial networks in [1798]. Approaches that rely on the validation set inherit the robustness of this method (up to any approximations), and avoid the need for knowing what kind of solutions are good a priori, but do introduce different procedures and datasets for weights and hyperparameters, breaking desideratum (2).

34.2.6 The Need for Something Better

In the past sections we have seen that several classes of methods have been developed with the aim of simplifying hyperparameter selection, and with various successes and compromises on our desiderata. Methods based on validation sets inherently rely on using different procedures for parameters and hyperparameters, which introduces complexity we would rather avoid (desideratum 2), while methods relying only on training data relied on parameters that needed to be tuned just right to make the method work (breaking desideratum 1). We argued that ultimately, empirical results are the only way to determine the success of an overall method.

For methods based on training sets only (Sections 34.2.3, 34.2.4), we saw that implicit regularisation could lead to selecting models that generalised well, even if the objective function itself did not prefer them. We argued that these implicit regularisers (initialisation, training dynamics, early stopping, using small model sizes) could be brittle and break when placed in a realistic scenario. The interaction between inductive bias and model size was a particularly strong example of this. Small model sizes could make the training loss prefer a constraining inductive bias, that did help generalisation performance. However, this effect can disappear when applied to models of sizes that are not artificially restricted, which is exactly the setting that deep learning cares about!

We draw two conclusions from this review. **Firstly**, that empirical evaluation of hyperparameter learning methods has potential pitfalls that we should be aware of. Empirical evaluations should be aware of implicit regularisation that may exist, and establish how widely this can be relied on. **Secondly**, we saw that every method broke some of our desiderata, or had failure modes.

The key trade-off came from the objective functions we considered. On the one hand, we had losses evaluated on validation sets, which robustly evaluated which methods generalised without relying on implicit or explicit regularisation, but which required awkward multi-stage optimisation processes. On the other hand, we had losses evaluated only on training sets, which were convenient to optimise, but which failed to distinguish between hyperparameters that performed differently, but which fit the data equally well.

In the following, we argue that Bayesian inference can provide us with a strong principle for developing procedures for learning hyperparameters.¹⁰ In particular, it leads to an objective function that *does* distinguish between models that generalise differently, while also satisfying our desiderata. The difficulty with the Bayesian approach is that it prescribes *what* should be computed to solve a particular problem without giving any hints on *how* this should be done. However, knowing what quantity to compute provides a helpful way to guide the development of methods, since efforts can focus on approximating this quantity accurately.

Currently, it is still an open problem for how to *accurately* approximate Bayesian inference in neural networks, despite much work that has already been carried out. We therefore review the capabilities we hope to achieve by continuing this line of work, and argue that it is worth to continue research in this direction.

¹⁰We do not argue that any (strictly) Bayesian approach is the *only* way to solve these problems! There are many other theories for assessing what models will generalise well without relying on validation data. While these methods are currently mostly used for explaining why deep learning works, they could also be investigated as principles for improving training procedures. It is also interesting to note that learning theorists focus on providing mathematically precise statements about generalisation. Bayesian inference does not. It only prescribes a procedure. Or alternatively, “Bayes tells us what to do, but not why it works”.

34.3 From Bayesian Hierarchical Inference to Model Selection

The general Bayesian procedure for inferring an unknown quantity is to put a prior on it, and then calculating the posterior distribution given the training data. One early hint that Bayes may lead us to a method that fits with our desiderata, is that *any* strictly Bayesian procedure only depends on *training data*, which was one of the requirements laid out in the introduction. Here, we review which quantities Bayesian inference *prescribes* us to compute, leaving concerns of how to compute practical approximations to later.

34.3.1 Uncertainty over the Weights: The Typical Bayesian Deep Learning Case

We begin with the problem that Bayesian deep learning is typically known for: quantifying uncertainty over the weights. We specify a model and its assumptions through its *probability of everything* [1111, p. 61], by specifying a prior on weights w and a likelihood given the data (x_n, y_n) :

$$p(\mathbf{y}, w | \theta, \mathbf{x}) \stackrel{\text{def}}{=} \prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta) \quad (34.12)$$

Typically, we have a normal distribution on weights, possibly with per-layer variances, with our likelihood depending only on the weights and hyperparameters through the neural network function $f_w^\theta(x_n)$, so we can write:

$$p(y_n | w, x_n, \theta) = p(y_n | f_w^\theta(x_n)), \quad p(w | \theta) = \prod_{\ell=1}^L \mathcal{N}(w_\ell; 0, \sigma_\ell^2). \quad (34.13)$$

We merge all hyperparameters into θ , so they control both the architecture, as well as the prior variances for each layer σ_ℓ^2 .

In the typical Bayesian deep learning case, hyperparameters are considered to be fixed, and they are typically found through trial-and-error/crossvalidation. This means that in order to make predictions, we can focus on finding the posterior on the weights:

$$p(w | \mathbf{x}, \mathbf{y}, \theta) = \frac{\prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta)}{p(\mathbf{y} | \theta, \mathbf{x})}. \quad (34.14)$$

Actually calculating a good approximation to this posterior is a significant challenge in itself, and many papers have been written on this topic.

34.3.2 Inferring Hyperparameters through Hierarchical Bayes

Bayes is a helpful starting point for solving inference problems, because it *prescribes* what should be computed to solve the problem. If we want to infer hyperparameters as well as weights, we can simply follow the previous procedure, and specify a “probability of everything” that includes the hyperparameters:

$$p(\mathbf{y}, w, \theta | \mathbf{x}) \stackrel{\text{def}}{=} \prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta) p(\theta). \quad (34.15)$$

To make predictions, all we need to do is find the posterior over both the weights w and hyperparameters θ . By grouping terms together, we can see that this posterior consists of the posterior over the weights from earlier (Eq 34.14) and an additional posterior over the hyperparameters:

$$p(w, \theta | \mathbf{y}, \mathbf{x}) = \frac{\prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta) p(\theta)}{p(\mathbf{y} | \mathbf{x})} \quad (34.16)$$

$$= \underbrace{\frac{\prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta)}{p(\mathbf{y} | \theta, \mathbf{x})}}_{=p(w | \mathbf{y}, \theta, \mathbf{x})} \cdot \underbrace{\frac{p(\mathbf{y} | \theta, \mathbf{x}) p(\theta)}{p(\mathbf{y} | \mathbf{x})}}_{=p(\theta | \mathbf{y}, \mathbf{x})}, \quad (34.17)$$

$$\text{with } p(\mathbf{y} | \theta, \mathbf{x}) = \int \prod_{n=1}^N p(y_n | w, x_n, \theta) p(w | \theta) dw. \quad (34.18)$$

The structure of this posterior is shared with that of any hierarchical model. Hierarchical modelling is a powerful technique that goes far back, and understanding its characteristics is very helpful in understanding why model selection gives benefits over non-hierarchical independent priors. Chapter 28 in MacKay [1111] gives an excellent overview of how to apply hierarchical modelling and Bayesian model selection, and the mechanisms by which it works (a measurement of simplicity, and an automatic Occam’s razor effect that removes unnecessary complexity). In-depth discussions can also be found in the statistics literature, e.g. chapter 5 and 15 of Gelman et al. [602] or Gelman [595]. Interestingly, there are different opinions on when hierarchical modelling is a valid thing to do, which we will discuss later. Nevertheless, in principle, Bayes gives a unified training procedure for weights and hyperparameters that also only depends on training data. As before, approximations are needed in practice, and research continues on how best to do this.

Leaving the *how* aside for the moment, we can already ask questions about how this *prescribed* procedure compares to the approaches discussed in Section 34.2, in terms of what information needs to be specified by the user. All terms that need to be specified are in Eq. 34.15. After taking the logarithm and comparing to Eq. 34.5, we notice that the likelihood $p(y_n|w, x_n, \theta)$ can be identified with the loss function, and $p(w|\theta)$ with the weight regulariser. Moreover, we can notice that the prior on the hyperparameters $p(\theta)$ can be identified with the explicit regulariser in Eq. 34.9.

The requirement to specify a loss and a regulariser is no more a-priori knowledge than what is required in “typical” deep learning. However, in Section 34.2.4 we criticised the need for explicit encouragement of solutions that were known to be good, which breaks desideratum (1). Can we determine that Bayes suffers from the same problem? To answer this question, we must investigate the hyperparameter posterior $p(\theta|\mathbf{y}, \mathbf{x})$ (Eq. 34.17). It is true that Bayes prescribes that we need to specify *some* prior $p(\theta)$, but the problem in Section 34.2.4 was that the hyperparameter regulariser was the *only* mechanism for preferring some solutions over others. In contrast, the posterior $p(\theta|\mathbf{y}, \mathbf{x})$ uses $p(\mathbf{y}|\theta, \mathbf{x})$ as the likelihood. We will discuss later (Section 34.4) that this behaves very differently to normal likelihoods, and includes an automatic Occam’s razor behaviour that also distinguishes models in a way that is related to their generalisation, allowing the method to work even with a flat prior $p(\theta)$.

34.3.3 Maximum Marginal Likelihood as a Practical Approximation

In both cases described above, we mentioned that approximations are needed. One general practical difficulty with applying Bayesian inference comes from needing to approximate analytically-intractable integrals in the posteriors and predictive distributions. *Any* approximation has to deal with the integration in one way or another.¹¹

Strictly speaking, Bayesian inference in hierarchical models *prescribes* us to make predictions using the full posterior over weights w and hyperparameters θ , which brings additional challenges over just computing posteriors over weights alone. The posterior over hyperparameters $p(\theta|\mathbf{y}, \mathbf{x})$ is often a lot harder to compute than of the level below in the hierarchy (in our case w). For example, in (hierarchical) Bayesian linear regression [1212, §11.6] or Gaussian processes [1382] the posterior that corresponds to w can be computed in closed form, while the posterior over hyperparameters cannot. More generally, integrating over the additional hyperparameter variables is harder than simply integrating over the weights w , and hierarchical structure makes this notoriously hard [146]. Neural networks share these properties, and moreover, integrating over multiple hyperparameters would mean keeping around multiple architectures, which is a large cost. Can Bayes still help to get adaptive architectures, without needing to directly ensemble over the whole architecture?

The *Maximum Marginal Likelihood* method, also known as *Type-II Maximum Likelihood*, suggests to represent only the mode of the posterior over *hyperparameters* $p(\theta|\mathbf{y}, \mathbf{x})$, instead of the whole distribution.¹² The key justification is that if the full hyperparameter posterior $p(\theta|\mathbf{y}, \mathbf{x})$ is highly peaked, integrals over it can be approximated by placing all its mass at its mode

$$p(\theta|\mathbf{y}, \mathbf{x}) \approx \delta(\theta - \theta^*) \quad (34.19)$$

$$\theta^* = \arg \max_{\theta} \log p(\mathbf{y}|\theta, \mathbf{x}) + \log p(\theta). \quad (34.20)$$

Additionally, if the prior on θ is flat, we can even disregard the hyperparameter prior $p(\theta)$ and seek out the maximum of the *marginal likelihood* only:

$$\theta^* = \arg \max_{\theta} \log p(\mathbf{y}|\theta, \mathbf{x}). \quad (34.21)$$

¹¹For discussions on the mathematical reasons for why this is hard, see MacKay [1111, §29.1], or Betancourt [144, §1] and its accompanying talk [145]).

¹²For examples in various models, see §7.5 in Murphy [1212], or §5.2 in Rasmussen and Williams [1382] for an application in Gaussian processes.

This approximation makes predictions with the usual weight posterior $p(w|\mathbf{y}, \mathbf{x}, \theta)$, but with the hyperparameters θ being set by the above point estimate. This still allows adaptability of the architecture, but without the need to consider multiple architectures.

The only quantity that is needed compared to the typical Bayesian deep learning case, is the marginal likelihood $p(\mathbf{y}|\theta, \mathbf{x})$. Interestingly, this quantity is computed directly when computing the weight posterior $p(w|\mathbf{x}, \mathbf{y}, \theta)$ (Eq. 34.14). In addition, approximate inference methods often also provide an estimate of the marginal likelihood as a byproduct of approximating the weight posterior, with the approximation quality of both being related (e.g. in variational inference [170] and annealed importance sampling [1228] a more accurate distribution approximation leads to a more accurate marginal likelihood). So, there may be reason to hope that this adaptivity can be gained without too much additional overhead compared to Bayesian learning of the weights alone.

So overall, we see that by starting from what Bayesian inference prescribes for inferring hyperparameters and applying a simple approximation, we arrive at a procedure that selects a single model. We can see that this procedure solves the same *model selection* problem as crossvalidation. Because of this, we will interchangeably refer to this procedure for selecting (mostly continuous) hyperparameters as Bayesian model selection, along with Maximum Marginal Likelihood, and Type-II Maximum Likelihood. This approximation has a long history, with early mentions of the phrase “type-II maximum likelihood” being in Good [648] and Berger [123].

The reasoning that we discussed so far followed Bayesian inference as a *prescriptive* procedure. We have simply followed the Bayesian prescription of solving inference problems by finding a posterior, and then applied one simplification to avoid the computational difficulty of integrating over the hyperparameters. The outcome of this path, is a procedure that can be directly applied to models where the weight posterior can be computed directly (e.g. linear models [1212, Chapter 9] or Gaussian processes [1382]), which is not the case for neural networks. And so, we will discuss further approximations in upcoming sections.

34.3.4 Practical Approximations

For Bayesian model selection we require the marginal likelihood $p(\mathbf{y} | \mathbf{x}, \theta)$ (or its log) and, more generally, the posterior $p(w | \mathbf{x}, \mathbf{y}, \theta)$. In neural networks these quantities are typically intractable because the likelihood $p(\mathbf{y} | \mathbf{x}, w, \theta)$ depends on w through a *nonlinear* mapping $w \mapsto f_w^\theta(\cdot)$ induced by the network (and because w is high-dimensional). As a result, the marginal likelihood (evidence) integral cannot be computed in closed form, motivating approximate inference.

Two closely related approximations are *variational inference* (VI) [170, 174, 660] and the *Laplace approximation* [1105]. Both typically approximate the posterior with a Gaussian distribution $q(w)$ and thereby induce a tractable approximation to the marginal likelihood. VI treats the mean and covariance of this Gaussian as variational parameters and optimises them through the evidence lower bound (ELBO), yielding a unified training criterion that can be differentiated with respect to both the parameters of $q(w)$ and the hyperparameters θ . The ELBO is a global, distribution level objective that balances expected data fit and prior complexity across $q(w)$. In contrast, Laplace restricts $q(w)$ to Gaussians centred at local MAP estimates, with covariance fixed by local posterior curvature. For the same Gaussian structure, the set of approximations attainable by Laplace is therefore a strict subset of the variational family available to VI. Consequently, the optimal VI ELBO is at least as high as the ELBO of any Laplace solution, and is strictly higher whenever the Laplace approximation is not itself ELBO optimal.

Regardless, the Laplace approximation can be very practical: it avoids solving an additional optimisation problem for $q(w)$ and instead constructs a Gaussian approximation from a point estimate (often the MAP weights) and local curvature information. This can make Laplace-based evidence approximations attractive when a good point estimate is readily available and when scalable curvature approximations are feasible.

In practice, both VI and Laplace often introduce additional structural approximations to make covariances (or precisions) computationally manageable at scale. Common choices include diagonal covariances, Kronecker-factored [812, 1136], or equivalently matrix-normal parameterisations [1083] in VI. These structural assumptions trade off fidelity and scalability and can be used within either approach: in VI they parameterise $q(w)$ directly, while in Laplace they parameterise or approximate the curvature used to form the Gaussian $q(w)$.

In literature, these approximations have provided differentiable surrogates of the marginal likelihood $\log p(\mathbf{y} | \mathbf{x}, \theta)$ for model selection and tuning hyperparameters θ (including symmetries) using only training data. Immer et al. [812] was the first to demonstrate Bayesian model selection at scale in large neural networks using the Laplace approximation, relying on Kronecker-factored approximations of the Generalized-Gauss Newton [191, 1136]. This was extended in [815] to allow differentiable Laplace-based evidence objectives for online learning of simple affine invariances using mini-batched training. In [1658] such scalable online Laplace approximations were used to learn layer-wise equivariance symmetries. A last-layer variational objective similar to deep kernel learning [1766] was

used for learning affine invariances in [1480, 1661], but the lower bound was only found tight enough to scale to single hidden-layer neural networks. In [1660], variational inference was applied to deeper networks for learning conservation laws and symmetries in Hamiltonian systems.

34.3.5 Applying Bayesian Model Selection

To get an idea of what capabilities Bayesian model selection can potentially bring to neural networks, we first illustrate its behaviour on a simple example where all the required calculations can be performed in closed form. We return to the example of Fig. 34.1 and consider training a model that can interpolate between two inductive biases as in Eq. 34.10: purely smooth, and periodic. We parameterise our space of functions using a feature map $\phi_\ell : \mathbb{R}^D \rightarrow \mathbb{R}^M$ consisting of many local “bump” basis functions arranged in a dense grid, with width ℓ .¹³ We use this to construct one space of functions $f_w^{\text{smooth}}(x)$ that just has a smoothness constraint, and another $f_w^{\text{periodic}}(x)$ which is also constrained to be periodic by first passing the input x through the map $u_p(x) = [\cos(2\pi x/p), \sin(2\pi x/p)]^T$. We specify our “probability of everything” as:

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad \theta = \{\ell_1, \ell_2, p, \theta_1, \theta_2, \sigma^2\} \quad (34.22)$$

$$f_{w,\theta}^{\text{smooth}}(x) = \phi_{\ell_1}(x)^T w_1, \quad (34.23)$$

$$f_{w,\theta}^{\text{periodic}}(x) = \phi_{\ell_2}(u_p(x))^T w_2, \quad (34.24)$$

$$f_w^\theta(x) = f_w^{\text{smooth}}(x) + f_w^{\text{periodic}}(x), \quad (34.25)$$

$$p(w|\theta) = \mathcal{N}\left(w; 0, \begin{bmatrix} e^{\theta_1} \cdot I_M & 0 \\ 0 & e^{\theta_2} \cdot I_m \end{bmatrix}\right), \quad (34.26)$$

$$p(y_i|w, \theta, x_i) = \mathcal{N}(y_i; f_w^\theta(x_i), \sigma^2) \quad (34.27)$$

We now have hyperparameters θ , which would need to be learned by crossvalidation if taking a point estimate for w to avoid the pitfalls of Sections 34.2.3, 34.2.4. Instead, we find a point estimate for θ according to Maximum Marginal Likelihood (Eq. 34.21) and find that after only a small number of datapoints, it strongly favours the periodic model (see Fig. 34.3).

We note several advantages over method that rely on (regularised) training fits:

- The marginal likelihood could distinguish between models that fit the data equally well, in contrast to the training loss (Fig. 34.1).
- Moreover, the marginal likelihood preferred the more constrained inductive bias, that allowed better generalisation on this problem.
- This preference came from the marginal likelihood directly, since we effectively used a flat prior $p(\theta)$. We did not need to use a prior $p(\theta)$ that encouraged inductive biases that were known to be good for the problem, as in Section 34.2.4.
- All hyperparameters were selected using the marginal likelihood, including those that controlled the weight prior $p(w|\theta)$.

In this example, Bayesian model selection provided a principle for finding a good inductive bias, and the right regularisation parameters that controlled the prior $p(w|\theta)$, while only relying on the training set. Moreover, it did not rely on imposing additional regularisers, that did not already exist in the typical deep learning formulation (Eq. 34.5). This demonstrates, in this simple example, capabilities that the other potential methods (Section 34.2) did not have, that get far closer to satisfying our three desiderata. The broader research goal is to get these benefits that exist for simple models, even in deep neural networks.

34.4 Why & When Bayesian Model Selection Works

Before we discuss practical approximations, we first return to the question that was raised at the end of Section 34.3.2: *Why* does this Bayesian procedure work? Bayes has told us *what* to do, but it has not explained *why* it

¹³We are following the weight-space parameterisation of Gaussian processes with Squared Exponential [1382, §4.2.1] and periodic [1113, 1382, §4.2.3] kernels. These are both single-layer fixed-basis function neural networks.

works. We can ask this question both about the exact procedure that Bayes has specified (Section 34.3.2) and the approximation we derived (Section 34.3.3). Answering this question will also provide insight into any limitations. In our discussions, we will take care to understand whether the problems we encounter are **a)** fundamentally hard problems about statistics (and so whether *any* method could solve it), **b)** difficulties with Bayesian inference in general, or **c)** problems that are introduced through the point estimate approximation of the full posterior $p(\theta|\mathbf{y}, \mathbf{x})$.¹⁴

As discussed in the previous section, Bayes prescribes that we find the hyperparameters through the posterior

$$p(\theta|\mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y}|\theta, \mathbf{x})p(\theta)}{p(\mathbf{y}|\mathbf{x})}, \quad (34.28)$$

and integrate them out when making predictions, with our practical approximation choosing the point estimate

$$\theta^* = \arg \max_{\theta} \log p(\mathbf{y}|\theta, \mathbf{x}). \quad (34.29)$$

In both cases, the key quantity that determines how the *data* influences the estimate is the *marginal likelihood* $p(\mathbf{y}|\theta, \mathbf{x})$. This is the key quantity that needs to be understood to understand both how the full hyperparameter posterior changes from the prior $p(\theta)$, and how our point estimate approximation behaves.

We can derive the marginal likelihood in terms of the model specification in the “probability of everything” (Eq. 34.15) as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \int \prod_{n=1}^N p(y_n|w, x_n, \theta)p(w|\theta)d\theta. \quad (34.30)$$

From this, we can immediately see that the integration over the global parameters $p(w|\theta)$ lead $p(\mathbf{y}|\mathbf{x}, \theta)$ to no longer be i.i.d. over $\mathbf{y}$, in contrast to the likelihood *given* the weights $p(\mathbf{y}|w, \mathbf{x}, \theta) = \prod_{n=1}^N p(y_n|w, x_n, \theta)$. It is this integration over the global parameters that gives the marginal likelihood its important Occam’s razor property, which is crucial for using it to select inductive hyperparameters that control inductive bias. The fact that it is a joint distribution is also the cause of it behaving fundamentally differently from methods that rely on the fit to the training set discussed in Sections 34.2.3 and 34.2.4. In the following, we investigate further properties of this distribution that explain why the marginal likelihood correlates more with generalisation than (regularised) training losses, where its Occam’s razor property comes from, and how it relates to other frameworks.

34.4.1 Losses During On-Line Prediction

One way to intuitively understand why the marginal likelihood is sensitive to how a model generalises, regardless of how it ends up fitting, is to apply the product rule of probability. By selecting an arbitrary ordering of our datapoints, we can rewrite the marginal likelihood as:

$$\log p(\mathbf{y}|\theta, \mathbf{x}) = \log p(y_1|x_1, \theta) + \log p(y_2|y_1, x_{2:n}, \theta) + \log p(y_3|y_{2:n}, x_{3:n}, \theta) + \dots \quad (34.31)$$

$$= \sum_{n=1}^N \log p(y_n|y_{1:n-1}, x_{n:n}, \theta). \quad (34.32)$$

We can view each term in the sum as the utility that is obtained when the model makes a prediction based on all $n - 1$ past datapoints, and the n th datapoint is the actual outcome, when the log probability is used as the utility function.¹⁵ The marginal likelihood is therefore the total utility obtained if this prediction procedure were played as all datapoints arrive in an on-line manner.¹⁶

This firstly explains why models that have the same fit to the training dataset can have vastly different marginal likelihoods. Models where past data predict future data well have high marginal likelihoods, whereas model fit only measures how a model memorises the training data. Secondly, this also makes a link to generalisation. Each term in the sum measures how well the model generalises on the n th datapoint. If the model has generalised well for many datapoints, the credibility that it will predict well on the next one increases too (see 34.4.3).

¹⁴In practical applications, the marginal likelihood itself will need to be approximated. This also introduces inaccuracies, and that gives an additional factor to be disentangled. A failure of an approximation to the marginal likelihood failing does not necessarily invalidate the maximum marginal likelihood method in general.

¹⁵With utility simply being the negative of loss, to keep the optimistic sign convention.

¹⁶See similar discussions in MacKay [1111, p. 353], and MacKay [1110].

We have visualised this process in Fig. 34.3 for the example discussed in Sections 34.2.1 and 34.3.5. We see that although the fit to the training data ends up being very similar for both models (right column), the marginal likelihood for the periodic model ends up being much higher, as the inductive bias allows for good predictions for unseen data with far less data than for the smooth model. This also indicates the better generalisation of the periodic model.

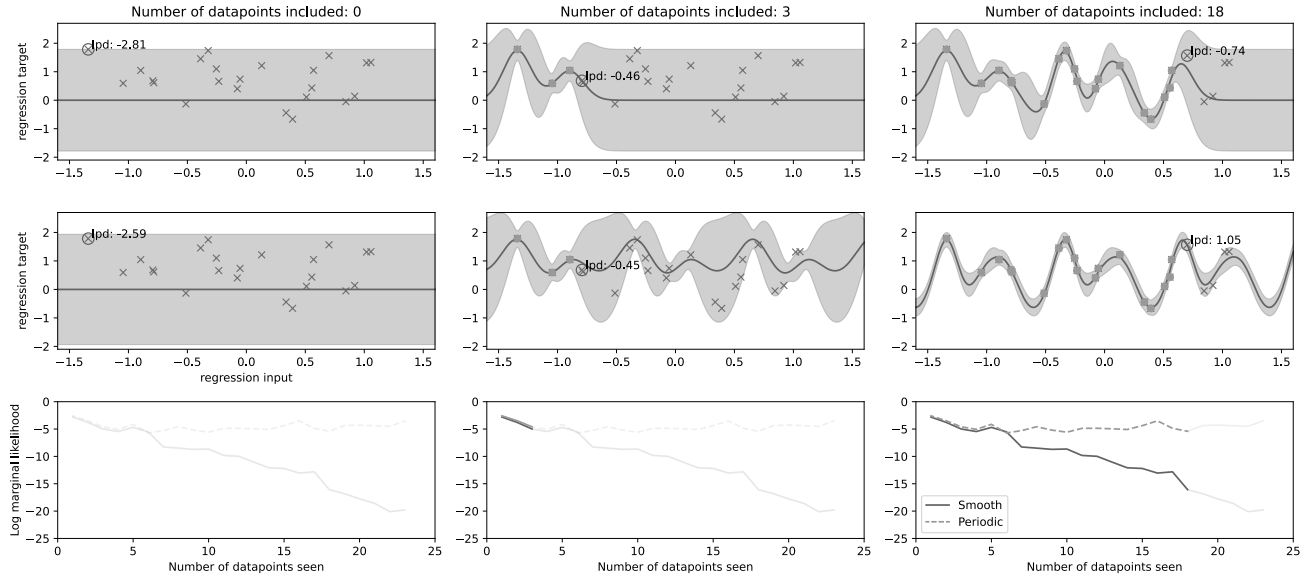


Figure 34.3: Regression predictions (with mean and 95% credible interval indicated) on a periodic dataset, with the smooth (top row) and periodic (middle row) models described in Section 34.3.5. All hyperparameters are optimised using the marginal likelihood, with the exception of the smooth model’s periodic component, which is clamped to zero. The columns differ in the number of datapoints given to the predictive posterior. The marginal likelihood is the sum of all predictive log likelihoods given past data points (indicated by a circle). The evolution of the marginal likelihood as more datapoints are added is plotted in the bottom row. For the periodic model, past parts of the dataset predict future parts well, and the model start predicting future datapoints well much earlier, indicating better generalisation.

34.4.2 Occam’s Razor

Another view on the marginal likelihood has been that it automatically embodies an “Occam’s razor” property that penalises model complexity. The key observation is that $p(\mathbf{y}|\theta, \mathbf{x})$ has to normalise over all possible datasets $\mathbf{y}$. Models that can fit many datasets must spread their probability mass thinly (and could be called more “complex”), while models that fit fewer datasets can assign higher probability to each (and could be called “simpler”). See Chapter 28 in MacKay [1111] or Rasmussen and Ghahramani [1380] for deeper discussions. An common illustration of this is depicted in Fig. 34.4, with [1214] providing a very helpful realistic depiction for a toy model.

34.4.3 Relationship to Other Frameworks: Crossvalidation, Generalisation Bounds, Compression

In addition to intuitive perspectives, other mathematical frameworks also investigate or prescribe the use of quantities that are similar to the marginal likelihood. The relationship to generalisation was made theoretically rigorous by Germain et al. [610] who noted that a monotonic relationship between a generalisation bound on samples from the posterior and the marginal likelihood of the model. However, this was under the constraint that no fitting of (hyper)parameters was done using the marginal likelihood, meaning that no generalisation bound for the Maximum Marginal Likelihood procedure (Section 34.3.3 is provided by this work. Nevertheless, the PAC-Bayesian framework can be extended to give a valid generalisation bound when hyperparameters are adapted, but this then required a full posterior over hyperparameters, *just as Bayes actually prescribes* (Section 34.3.2). This serves as a reminder that Maximum Marginal Likelihood is justified by the Bayesian framework only as an approximation when the posterior is peaked. When it is not, Bayes prescribes that hyperparameters are integrated.

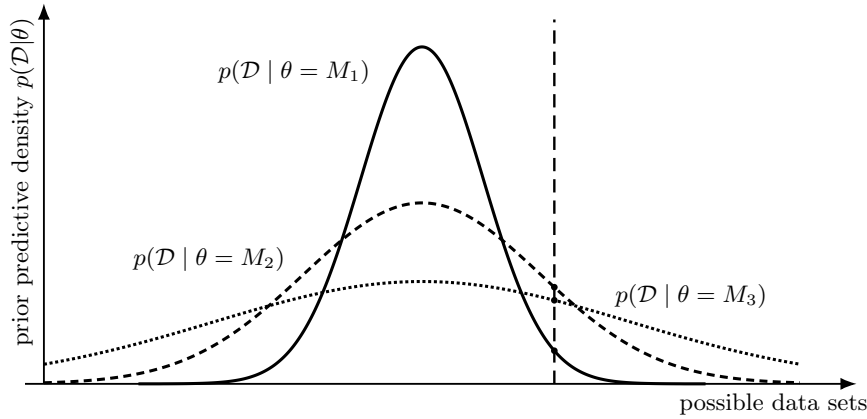


Figure 34.4: Cartoon illustrating that models must have distributions over datasets $p(\mathbf{y}|\theta, \mathbf{x})$ that normalise. Simple models spread their mass narrowly, and can give high marginal likelihoods to a small set of datasets, while complex models spread their mass widely, but cannot assign high marginal likelihood to all of these datasets. This property of the marginal likelihood is the mechanism for automatically penalising model complexity.

Another perspective is that the log marginal likelihood $\log p(\mathbf{y}|\theta, \mathbf{x})$ actually quantifies the coding length of how many bits are required to encode both the training data $\mathbf{y}$ and the model parameters $\mathbf{w}$. Compression is key to generalisation [173], and the Minimum Description Length (MDL) community [676, 1399, 1698] has argued for selecting models based on which compresses most. In MDL terminology, the marginal likelihood corresponds to a one-part Bayesian mixture code, and the sequential decomposition in Section 34.4.1 corresponds to a *prequential* code. Again, just like the Bayesian prescription, and the PAC-Bayesian analysis, MDL also notes that maximising the marginal likelihood w.r.t. hyperparameters is an approximation, given that it ignores the coding cost of the hyperparameters themselves. Again, the solution is to introduce a posterior over hyperparameters.

Finally, there is also a relationship between the marginal likelihood with the model selection method that we consider our “gold standard”. Fong and Holmes [537] showed that the marginal likelihood is equivalent to a crossvalidation scheme that averaged over all splits between training and testing sets. This correspondence shows that there is nothing magical about the marginal likelihood, in that it has equivalences with established methods, and therefore shares strengths and weaknesses. One key benefit of this mathematical relationship, is that formulation of the marginal likelihood as an integral may actually allow for approximations that are relatively computationally efficient. Crossvalidation is actually an incredibly expensive method, that requires retraining a whole machine learning method many times. While evaluating the integral of the marginal likelihood is certainly not easy, it does open up the possibility for applying a whole host of approximation techniques that are not available when viewing crossvalidation the “retrain-evaluate” perspective.

34.5 Problems with Bayesian Model Selection

Bayesian model selection in general, and maximising the marginal likelihood in particular have been discussed many times over the years. Sometimes, the critiques have even strongly argued against the use of this method. In many ways, our exposition in previous sections has followed exactly what Bayes straightforwardly prescribes, and which has been explained by many researchers who came before us. We take the position that this framework is mostly correct, and that any problems with the Bayesian framework as a whole, and Maximum Marginal Likelihood specifically, are small compared to the unrealised benefits that they offer. The challenge is in how compute the required quantities, and how to specify good models. Whether this is possible is an open research problem, with at most indirect evidence for and against.

Nevertheless, it is good to be aware of the limitations of the framework that we base our methods on. Three limitations stand out.

34.5.1 Sensitivity to Weight Prior

The marginal likelihood has its name because it is found by marginalising over the parameters of a model (Eq. 34.30), so its value certainly depends on the choice of the weight prior. The problem is that once the dataset is large enough, the predictions of the posterior may depend only weakly on the prior, while the value

of the marginal likelihood is very sensitive to it. This means that there exist ways to select priors for models so that they predict well, but have a poor marginal likelihood. This may cause worse models to be selected. Gelman [598] and Mackay had a frequently cited discussion on whether the Occam’s razor behaviour of Bayesian model selection and the interplay with priors was desirable or not. The resolution we subscribe to follow the explanation of Occam’s razor of MacKay [1111], but to follow Gelman’s advice on how to *apply* Bayesian model selection. In short, Gelman [598] argues against using Bayesian model selection over a discrete set of models. Instead, Gelman [594] argues for “bridging” these models using some continuous parameterisation. We followed exactly this advice in Section 34.3.5, where we allowed a continuous interpolation between the two extreme inductive biases of only smoothness or full periodicity. We follow this same advice in our later architecture search.

Lotfi et al. [1081] also raise the issue with the prior, and argue that using highly dispersed priors can really hurt the marginal likelihood, while still generalising well. The paper includes experiments where changing the variance of the weigh prior $p(w|\theta)$ can break correlation between the marginal likelihood and generalisation. This effect certainly exists, and can cause the selection of a sub-optimal model. Luckily, properties of the prior can be adjusted using the marginal likelihood (see Section 34.3.5 for an example), so long as the number of hyperparameters does not grow so large that the point estimate approximation becomes too poor. In our work on hyperparameter selection, this is how we deal with this problem.

Lotfi et al. [1081] suggest a “conditional log marginal likelihood” as an alternative solution, which removes the first few terms in the log marginal likelihood sum of Eq. 34.32. This can help remove some prior dependence, but it comes at the price of also needing to tune how many terms to remove, while also introducing another split within the training set, both of which we are aiming to avoid according to our desiderata.

34.5.2 Overfitting when Fitting Many Hyperparameters

Increasing the number of parameters that are to be estimated using the marginal likelihood also comes with problems. It is well-known [1382, p. 109] that overfitting can even occur for the marginal likelihood when too many hyperparameters are added relative to the dataset size (e.g. when putting all neural network parameters in the hyperparameters [1253]). This behaviour is also discussed by more recently Lotfi et al. [1081] as an argument against the marginal likelihood.

When assessing this problem, it is important to keep two things in mind. Firstly, that there are limits to how many models we can select between from a finite data. Even crossvalidation can end up overfitting in this way [248, 1234]. Secondly, that there are ways to mitigate against these problems. The Bayesian prescription (Section 34.3.2) is to fall back on the prior when the marginal likelihood is not peaked enough, and that we need to perform integration over hyperparameters. Another solution is to be careful to not include too many hyperparameters to optimise over, and leave it as an empirical question to formulate a procedure that can be shown to be generally applicable. We take the latter approach.

34.5.3 Architectures are Defined

One final criticism about this approach, is the degree to which we are really *learning* inductive biases. In all our approaches, we always have to specify a set of inductive biases to select from. In our example from section 34.3.5, our model could select various amounts of periodicity. One objection that could be raised is that the method only worked because the inductive bias (e.g. periodicity) was built in! If the dataset did not contain periodicity (or other properties that fit an inductive bias in our set), the inductive bias cannot be learned.

Again, this is true. However, there are two responses to this. Firstly, we must compare against the alternative: not adapting inductive biases. In the example from section 34.3.5, we would be stuck with just using the smooth function. With hyperparameter adaptation, we can gain the advantage of an inductive bias in our set (e.g. periodicity) if we are lucky enough to encounter dataset that is suitable for it. If the dataset at hand does not have exploitable structure, we are no worse off. Secondly, we should keep in mind that every machine learning method must specify a candidate set of solutions that one is chosen from. Neural networks themselves parameterise spaces of functions. If the function that is to be learned is outside this space, it cannot be learned. Developing a space of inductive biases is an active research direction, and geometric deep learning [210] is a rich area that is exploring this space. Developing Bayesian model selection further would be a good counterpart to geometric deep learning research, since it would allow automatically selecting an inductive bias within this increasingly rich set.

34.6 Exact symmetry learning in basis function model

To make the role of marginal likelihood concrete, we begin with a tractable shallow model: Bayesian linear regression with fixed basis functions. This allows us to compute the marginal likelihood in closed form and to see explicitly how hyperparameters θ can shape the inductive bias. As discussed in the previous section, the symmetry is not embedded explicit prior on weights, but in the parameterization of the function class, here through the choice of basis functions. We assume inputs $x \in \mathbb{R}^{D_{\text{in}}}$ and a vector of basis functions $\phi(x) \in \mathbb{R}^M$. The model is

$$y = w^\top \phi(x) + \epsilon, \quad (34.33)$$

where $w \in \mathbb{R}^M$ are weights and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We place a standard isotropic Gaussian prior on the weights:

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I), \quad (34.34)$$

with precision hyperparameter $\alpha > 0$. Given data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, the likelihood is

$$p(y \mid w, \sigma^2) = \mathcal{N}(y \mid \Phi w, \sigma^2 I), \quad (34.35)$$

with design matrix $\Phi = [\phi(x_1)^\top; \dots; \phi(x_N)^\top]$. Because both prior and likelihood are Gaussian, the marginal likelihood has a closed form [160, 1105, 1382]:

$$p(y \mid \alpha, \sigma^2) = \int p(y \mid w, \sigma^2) p(w \mid \alpha) dw \quad (34.36)$$

$$= \mathcal{N}(y \mid 0, \sigma^2 I + \frac{1}{\alpha} \Phi \Phi^\top). \quad (34.37)$$

The corresponding log marginal likelihood is

$$\log p(y \mid \alpha, \sigma^2) = -\frac{1}{2} y^\top C^{-1} y - \frac{1}{2} \log |C| - \frac{N}{2} \log(2\pi), \quad (34.38)$$

with covariance

$$C = \sigma^2 I + \frac{1}{\alpha} \Phi \Phi^\top. \quad (34.39)$$

Here the hyperparameters θ correspond to choices in the basis functions $\phi(x; \theta)$, which parameterize the prior over functions. By maximizing the marginal likelihood,

$$(\hat{\theta}, \hat{\alpha}, \hat{\sigma}^2) = \arg \max_{\theta, \alpha > 0, \sigma^2 > 0} \log p(y \mid \alpha, \sigma^2, \theta), \quad (34.40)$$

we jointly learn the symmetry parameter, weight-prior precision, and observation-noise variance from the marginal likelihood. Additional basis hyperparameters, such as length scales, can be included in the same optimization. Thus, even in this shallow example, marginal likelihood provides a clear, analytic demonstration of how inductive bias, encoded via parameterization of the basis functions, can be selected in a principled way.

34.6.1 Example: Flip symmetry.

Consider one-dimensional inputs $x \in \mathbb{R}$ and let $\phi(x)$ consist of basis functions, such as radial basis functions RBFs [708], centered at a set of locations with fixed length scales. A flip symmetry $x \mapsto -x$ can be embedded by symmetrizing the basis (through relaxed group averaging):

$$\phi'(x; \theta) = (1 - \theta) \phi(x) + \theta \frac{1}{2} [\phi(x) + \phi(-x)], \quad (34.41)$$

where $\theta \in [0, 1]$ controls the degree of symmetry. For $\theta = 0$, the basis is unconstrained; for $\theta = 1$, the basis is fully flip-symmetric; and intermediate values interpolate between the two. In this way, θ plays the role of the hyperparameter that governs symmetry in the prior over functions. Equivalently, decomposing the original basis into its symmetric and antisymmetric components gives

$$\phi'(x; \theta) = \phi_{\text{sym}}(x) + (1 - \theta) \phi_{\text{asym}}(x), \quad (34.42)$$

and therefore

$$\phi'(x; \theta) - \phi'(-x; \theta) = (1 - \theta) [\phi(x) - \phi(-x)]. \quad (34.43)$$

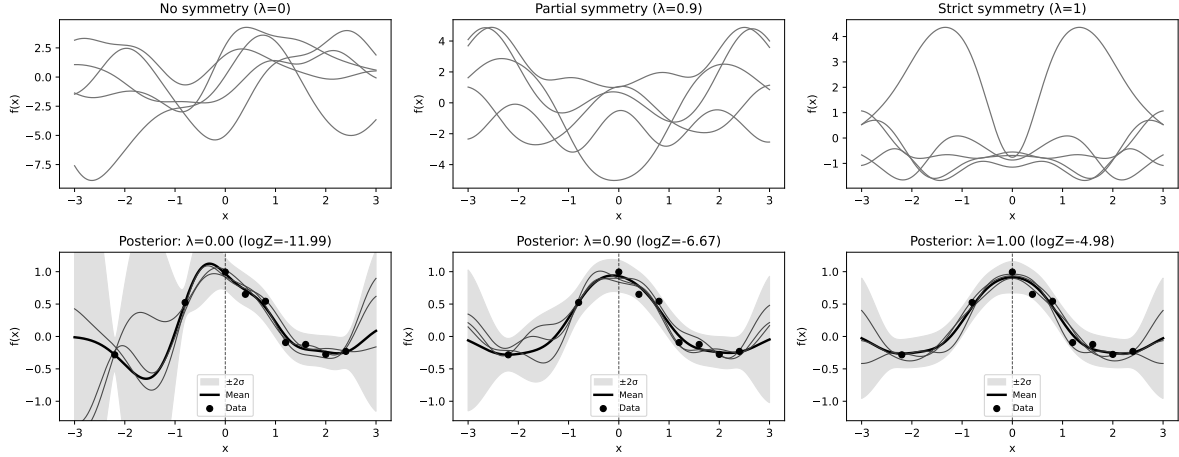


Figure 34.5: Priors and posteriors as θ suppresses antisymmetric variation; exact flip symmetry occurs at $\theta = 1$.

Thus, $\theta = 1$ enforces exact flip symmetry, while intermediate values continuously shrink, but do not eliminate, antisymmetric variation.

Figure 34.5 shows how the prior over functions and resulting posteriors change as θ increases. Larger θ values increasingly concentrate the function prior near the flip-symmetric subspace by shrinking its antisymmetric component. Exact symmetry is enforced only at $\theta = 1$, whereas smaller values retain varying degrees of asymmetric flexibility. The choice of θ is therefore central to determining how strongly symmetry is expressed in the posterior.

As illustrated in Figure 34.6, the marginal likelihood favors different degrees of symmetry for the two displayed datasets. For the symmetric case, the evidence curve favors large θ , and the posterior mean is approximately flip-symmetric. For the asymmetric case, the evidence peaks near $\theta = 0$, allowing the posterior to retain asymmetric variation. Taken together, these figures demonstrate that even in this simple setting, the marginal likelihood provides a principled mechanism for inferring whether symmetries should be imposed, and to what degree. Instead of manually building the inductive bias into the architecture or basis—which can improve generalisation when the bias is correct—the marginal likelihood can select the appropriate symmetry strength from the training data.

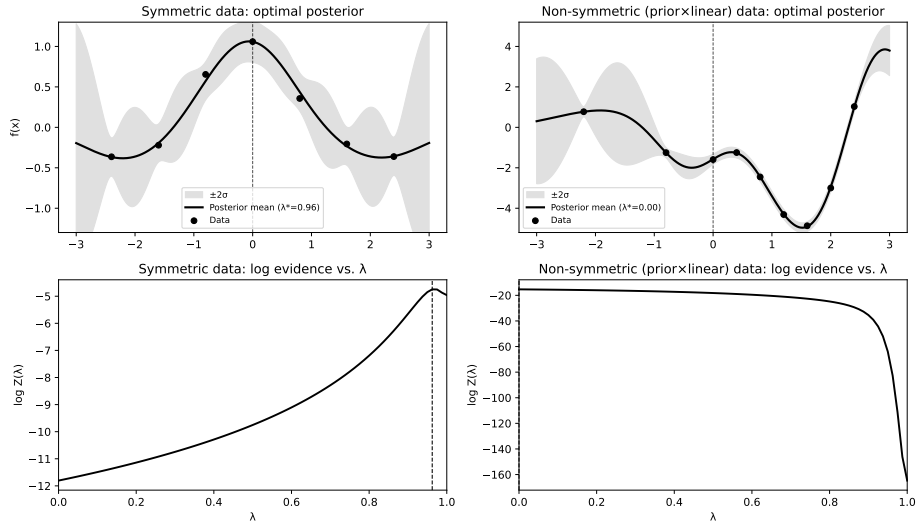


Figure 34.6: Effect of θ on posterior inference and the log marginal likelihood. **Left:** symmetric data. **Right:** asymmetric data. Top: data and posterior means at the evidence-optimal θ . Bottom: $\log Z(\theta)$.

34.7 Learning invariance in a shallow network

We now move beyond closed-form basis-function models to shallow neural-network classifiers, for which the non-conjugate likelihood makes the marginal likelihood intractable. We therefore use a variational bound whose complexity-sensitive objective can guide both weight and symmetry learning [1661]. The construction also connects group averaging of inputs to transformations and sharing of first-layer weights, as used in convolutional architectures [318, 914, 994].

34.7.1 Transformation-averaged function construction

We begin with a non-invariant base network

$$g_w : \mathbb{R}^{D_{\text{in}}} \rightarrow \mathbb{R}^K, \quad (34.44)$$

parameterised by w . A straightforward way to achieve (approximate) invariance is to average its outputs over a learnable transformation distribution $p_\theta(T)$:

$$f_w^\theta(x) = \mathbb{E}_{T \sim p_\theta(T)}[g_w(Tx)]. \quad (34.45)$$

Concentration at the identity recovers g_w , while coverage of more relevant transformations, rather than merely larger coordinate variance, typically promotes stronger approximate invariance. Strict invariance is the special case in which $p_\theta(T)$ is uniform over the relevant finite or compact group (its normalised Haar measure), so that the function remains exactly unchanged under further group actions.

Combined with a weight prior $p(w)$, this construction induces the function prior

$$p(f \mid \theta) = \int \delta(f - f_w^\theta) p(w) dw, \quad (34.46)$$

so θ controls the type and strength of invariance. In practice, the transformation integral is approximated by

$$f_w^\theta(x) \approx \frac{1}{M} \sum_{m=1}^M g_w(T^{(m)}x), \quad T^{(m)} \sim p_\theta(T). \quad (34.47)$$

Gradient-based learning of the inductive-bias hyperparameters requires both a differentiable marginal-likelihood objective and a differentiable sampling procedure for $T \sim p_\theta(T)$. The next section constructs such a procedure for affine transformations.

34.7.2 Example: affine transformations of 2D images

To make symmetry learning concrete, we parameterise local transformations from the affine group acting on 2D images. Using infinitesimal *generators* $G_i \in \mathbb{R}^{3 \times 3}$, a transformation is represented by the 3×3 homogeneous coordinate matrix

$$A(\alpha) = \exp\left(\sum_{i=1}^6 \alpha_i G_i\right) = \begin{pmatrix} a_{11}(\alpha) & a_{12}(\alpha) & t_1(\alpha) \\ a_{21}(\alpha) & a_{22}(\alpha) & t_2(\alpha) \\ 0 & 0 & 1 \end{pmatrix}. \quad (34.48)$$

Here, $\alpha = (\alpha_1, \dots, \alpha_6)$ contains generator coordinates for rotation, scaling, shear, and translation. The matrix entries $a_{jk}(\alpha)$ and $t_j(\alpha)$ are the resulting functions of these coordinates after exponentiation. To obtain a bounded distribution over transformations that matches the range parameterisation used in the experiments, we sample

$$\epsilon_i \sim \text{Uniform}(-1, 1), \quad (34.49)$$

$$\alpha_i = \theta_i \epsilon_i, \quad \theta_i \geq 0. \quad (34.50)$$

The parameter θ_i is therefore the maximum magnitude of the coefficient associated with generator G_i . Using the *reparameterisation trick* [892], the sampled transformation is

$$A(\theta, \epsilon) = A(\alpha(\theta, \epsilon)) = \exp\left(\sum_{i=1}^6 \theta_i \epsilon_i G_i\right). \quad (34.51)$$

When $\theta = 0$, the transformation is the identity. Increasing a component θ_i expands the range of transformations along the corresponding generator, typically promoting stronger approximate invariance in that direction. This construction allows gradients to flow through both w and θ , enabling joint optimisation of the network parameters and the invariance parameters. It is commonly used in invariance learning [120, 815, 1661].

The matrix $A(\alpha)$ transforms pixel coordinates in homogeneous form. Writing $u = (u_1, u_2)$ for a pixel location and $\tilde{u} = [u_1, u_2, 1]^\top$, the induced action on an image x is the pullback

$$(T_\alpha x)(u) = x(\pi(A(\alpha)^{-1}\tilde{u})), \quad (34.52)$$

where $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ returns the first two coordinates. Off-grid arguments are evaluated by bilinear or bicubic interpolation. For fixed α , this interpolation is linear in x , so T_α may be represented as a matrix in $\mathbb{R}^{D_{\text{in}} \times D_{\text{in}}}$; discretisation makes the group relations only approximate.

The same construction has a weight-space reading. Let $W_1 \in \mathbb{R}^{H \times D_{\text{in}}}$ be the first-layer weight matrix, so the linear map is $x \mapsto W_1 x$. Identifying T with its matrix representation in $\mathbb{R}^{D_{\text{in}} \times D_{\text{in}}}$,

$$W_1(Tx) = (W_1 T)x, \quad (34.53)$$

and therefore

$$\mathbb{E}_{T \sim p_\theta(T)} [W_1(Tx)] = \left(\mathbb{E}_{T \sim p_\theta(T)} [W_1 T] \right) x. \quad (34.54)$$

Averaging over input transformations is thus equivalent to averaging the orbit of transformed first-layer weights $\{W_1 T\}$. For translations, that orbit is implemented by convolution followed by pooling [318, 994]; Section 34.9.1 revisits this connection in the context of learning equivariances.

34.7.3 Variational objective for a shallow network

We now turn from constructing the transformation-averaged model to learning its invariance parameters. For classification we apply a softmax to the averaged logits,

$$p(y \mid x, w, \theta) = \text{softmax}(f_w^\theta(x))_y, \quad (34.55)$$

and select θ by maximising the marginal likelihood $\log p(\mathcal{D} \mid \theta)$, as in the basis-function setting. Because this integral is intractable for neural classifiers, we use a last-layer variational bound: a posterior over the final-layer weights, with the first-layer weights and invariance parameters treated as deterministic hyperparameters [1661].

Concretely, take the shallow logit network

$$g_w(x) = W_2 \sigma(W_1 x), \quad (34.56)$$

with the same transformation average $f_w^\theta(x) = \mathbb{E}_{T \sim p_\theta(T)} [g_w(Tx)]$. Treating W_1 as a model hyperparameter and placing a prior only on W_2 yields the conditional function prior

$$p(f \mid W_1, \theta) = \int \delta(f - f_{W_1, W_2}^\theta) p(W_2) dW_2 \quad (34.57)$$

and marginal likelihood

$$p(y \mid X, W_1, \theta) = \int p(y \mid X, W_1, W_2, \theta) p(W_2) dW_2. \quad (34.58)$$

We use a Gaussian prior $p(W_2)$ and a Gaussian variational posterior $q_\Omega(W_2)$, with a separate covariance block for each output class, while optimising W_1 and θ as deterministic hyperparameters. The transformation-exact ELBO is

$$\mathcal{L}_{\text{exact}}(W_1, \theta, \Omega) = \sum_{n=1}^N \mathbb{E}_{q_\Omega(W_2)} \left[\log \text{softmax}(\mathbb{E}_{T \sim p_\theta(T)} [g_w(Tx_n)])_{y_n} \right] \quad (34.59)$$

$$- \text{KL}(q_\Omega(W_2) \parallel p(W_2)). \quad (34.60)$$

For a finite number M of transformation samples, define

$$\mathcal{L}_M(W_1, \theta, \Omega) = \sum_{n=1}^N \mathbb{E}_{\substack{W_2 \sim q_\Omega(W_2), \\ T_n^{(m)} \stackrel{\text{iid}}{\sim} p_\theta(T), m=1, \dots, M}} \left[\log \text{softmax} \left(\frac{1}{M} \sum_{m=1}^M W_2 \sigma(W_1 T_n^{(m)} x_n) \right)_{y_n} \right] \quad (34.61)$$

$$- \text{KL}(q_\Omega(W_2) \parallel p(W_2)). \quad (34.62)$$

Model	Fully rotated MNIST	Test accuracy (%)		Original MNIST
		Translated MNIST	Scaled MNIST	
MLP	90.35	89.34	96.61	98.10
MLP + Rotation (VI)	98.05	94.08	97.62	98.64
MLP + Translation (VI)	93.59	97.87	97.98	98.76
MLP + Scale (VI)	93.80	94.30	98.06	98.35
MLP + Affine (VI)	98.14	97.66	98.31	98.93

Table 34.1: Test accuracy (%) of shallow ReLU multilayer perceptrons (MLPs) with different candidate invariance families on four MNIST variants. Adapted from van der Ouderaa and van der Wilk [1661].

Since $\log \text{softmax}(z)_y$ is concave in z , Jensen’s inequality gives

$$\mathcal{L}_M(W_1, \theta, \Omega) \leq \mathcal{L}_{\text{exact}}(W_1, \theta, \Omega) \leq \log p(y \mid X, W_1, \theta). \quad (34.63)$$

The first inequality is the finite-transformation Jensen gap, whereas the second is the variational gap. For $M = 1$, this recovers the single-transformation bound; under the usual integrability conditions, $\mathcal{L}_M$ converges to $\mathcal{L}_{\text{exact}}$ as $M \rightarrow \infty$. Optimising W_1 jointly with θ and Ω corresponds to empirical Bayes optimisation of the deterministic model hyperparameters alongside approximate inference over W_2 .

Let $W_2^{(\ell)} \sim q_\Omega(W_2)$ and $T_n^{(\ell, m)} \sim p_\theta(T)$. An unbiased estimator of $\mathcal{L}_M$ based on L weight samples is

$$\hat{\mathcal{L}}_M(W_1, \theta, \Omega) = \frac{1}{L} \sum_{\ell=1}^L \sum_{n=1}^N \log \text{softmax} \left(\frac{1}{M} \sum_{m=1}^M W_2^{(\ell)} \sigma(W_1 T_n^{(\ell, m)} x_n) \right)_{y_n} \quad (34.64)$$

$$- \text{KL}(q_\Omega(W_2) \parallel p(W_2)). \quad (34.65)$$

This estimator provides a practical stochastic objective for joint training. With minibatches, the sum over data points is replaced by a suitably rescaled minibatch sum. Reparameterising samples from $q_\Omega(W_2)$ and $p_\theta(T)$ provides pathwise gradients for Ω and θ , while W_1 is differentiated directly. Thus, W_1 , Ω , and θ can be optimised jointly using standard stochastic-gradient methods. Increasing L reduces estimator variance, while increasing M improves the finite-transformation approximation. This is the objective used in the experiments below.

34.7.4 Experiments: learning useful invariances

We now evaluate whether the variational objective (i) improves predictive performance, (ii) learns interpretable transformation distributions, and (iii) provides a useful symmetry-learning signal beyond point-estimate cross-entropy.

Predictive performance

We compare learned-invariance models with a non-invariant multilayer perceptron (MLP). Each model is trained separately on fully rotated, translated, scaled, and original MNIST; Table 34.1 reports the resulting test accuracies (see 1661 for experimental details and corresponding CIFAR-10 results).

The learned-invariance models outperform the baseline in every setting. The translation-specific model performs best on translated MNIST, while the affine model performs best on the rotated, scaled, and original variants and has the highest average accuracy.

Inspecting learned transformations

The weight-space view also lets us inspect what the model has learned. Figure 34.7 shows samples from the first-layer filter orbits introduced above; variation within each orbit reflects the transformation distribution $p_\theta(T)$ learned for that dataset.

Role of the variational objective

Finally, we isolate the role of the training objective. We compare the VI-based model with the same transformation-averaged architecture trained using point-estimate cross-entropy: $q(W_2)$ is replaced by a point estimate, W_1 and

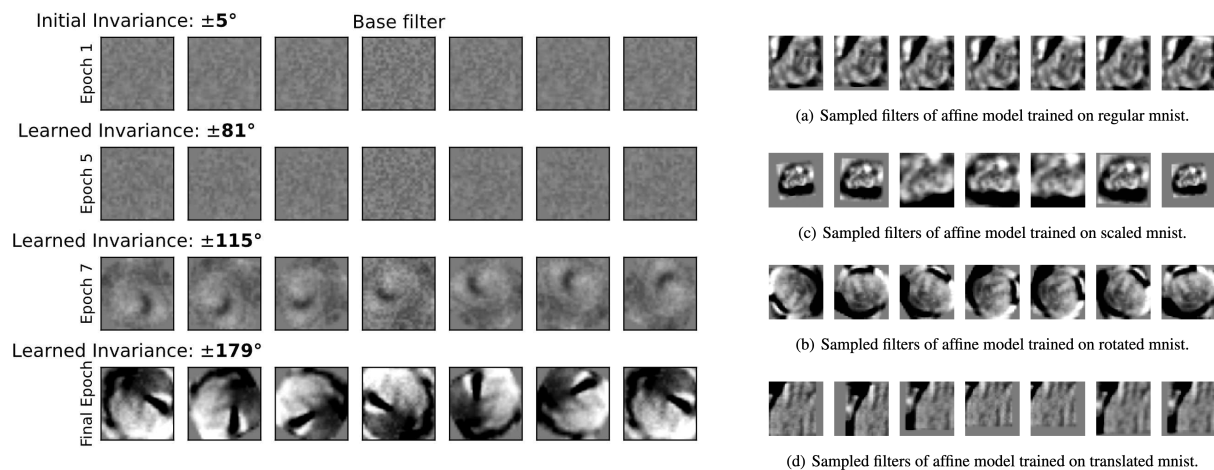


Figure 34.7: Filter-bank samples obtained by transforming the rows of W_1 under the learned distribution $p_\theta(T)$. Adapted from [1661].

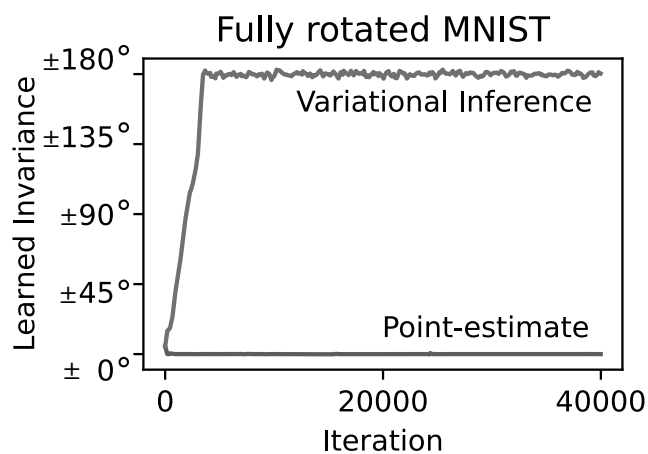


Figure 34.8: Predicted invariance over training iterations on fully rotated MNIST under point-estimate cross-entropy and variational inference. Adapted from [1661].

W_2 are optimised directly, and the KL term is removed. Figure 34.8 tracks the predicted invariance during training on fully rotated MNIST.

Point-estimate cross-entropy fails to recover the target invariance in this experiment, whereas the VI-based approach succeeds. This does not imply that maximum likelihood can never learn symmetry. Rather, when a sufficiently flexible network already fits the training data, the likelihood may provide little signal for selecting the invariance parameter. The variational objective adds a complexity-sensitive signal through posterior averaging and the KL term.

Together, these experiments show that the variational objective can learn useful, interpretable invariances in a shallow network. In the next chapter, we retain the same transformation-averaged construction but replace last-layer VI with a scalable Laplace approximation to allow invariances to be learned through approximate Bayesian model selection in *deep* networks.

34.8 Learning invariances in deep neural networks

We now turn to learning invariances in *deep neural networks*. The overall construction of obtaining (approximate/learnable) invariance through group averaging is unchanged, but integrating over all weights to obtain the marginal likelihood requires a scalable approximation. We use a Laplace approximation together with structured curvature approximations that make learning the invariance parameters θ tractable in deep architectures [812, 1402].

34.8.1 Differentiable marginal likelihood via the Laplace approximation

The shallow model combined transformation-averaged logits and a finite-sample Jensen bound with last-layer variational inference. For deep networks, we retain the same likelihood construction but integrate over all weights w under a prior $p(w \mid \lambda)$, where λ denotes its hyperparameters. With S transformation samples per input,

$$\hat{f}_{S,w}^\theta(x_n) = \frac{1}{S} \sum_{s=1}^S g_w(T_n^{(s)} x_n), \quad T_n^{(s)} \stackrel{\text{iid}}{\sim} p_\theta, \quad (34.66)$$

and the finite- S expected log likelihood is

$$\ell_S(w, \theta) = \sum_{n=1}^N \mathbb{E}_{T_n^{(1:S)}} \left[\log \text{softmax} \left(\hat{f}_{S,w}^\theta(x_n) \right)_{y_n} \right]. \quad (34.67)$$

By the same concavity argument as in the shallow model, ℓ_S lower-bounds the exact transformation-averaged log likelihood pointwise in w and converges to it as S increases.

This finite- S bound induces the evidence surrogate

$$Z_S(\theta, \lambda) = \int \exp(\ell_S(w, \theta)) p(w \mid \lambda) dw \leq p(y \mid X, \theta, \lambda). \quad (34.68)$$

In practice, neural-network optimisation selects a local mode of the finite- S surrogate log joint and evaluates the posterior curvature there:

$$\hat{w}_S(\theta, \lambda) \in \underset{w}{\text{local argmax}} [\ell_S(w, \theta) + \log p(w \mid \lambda)], \quad (34.69)$$

$$H_S = \nabla_w^2 [-\ell_S(w, \theta) - \log p(w \mid \lambda)] \big|_{w=\hat{w}_S}. \quad (34.70)$$

Treating this mode as dominant and assuming its curvature is positive definite, the single-mode *Laplace approximation* [815] gives

$$\log Z_S(\theta, \lambda) \approx \ell_S(\hat{w}_S, \theta) + \log p(\hat{w}_S \mid \lambda) - \frac{1}{2} \log |H_S| + \frac{P}{2} \log(2\pi), \quad (34.71)$$

where $P = \dim(w)$. A positive prior precision or explicit damping ensures positive definiteness for the curvature approximation used below. Although $\log Z_S$ lower-bounds the exact marginal likelihood, the subsequent Laplace and curvature approximations need not preserve this bound. Together, however, they provide a tractable differentiable approximation to the marginal likelihood that can be used to learn invariance and prior hyperparameters in deep networks.

34.8.2 Scalable curvature approximations

The remaining computational bottleneck is the curvature term $\log |H_S|$. We approximate it in two stages: first replacing the exact likelihood Hessian by the Generalized Gauss–Newton (GGN) matrix, and then imposing Kronecker-factored structure on each layer block.

GGN for transformation-averaged logits

The GGN drops second derivatives of the network outputs and retains the positive-semidefinite term $J^\top H_L J$ [191]. For one sampled set of transformations, let $J_{n,s} = J_g(T_n^{(s)} x_n; \hat{w}_S)$ and define the averaged Jacobian

$$\bar{J}_n = \left. \frac{\partial \hat{f}_{S,w}^\theta(x_n)}{\partial w^\top} \right|_{w=\hat{w}_S} = \frac{1}{S} \sum_{s=1}^S J_{n,s}, \quad (34.72)$$

where J_g is the Jacobian of the base-network logits. The sampled GGN likelihood curvature is

$$\hat{H}_{\text{GGN},S} = \sum_{n=1}^N \bar{J}_n^\top H_{L,n} \bar{J}_n, \quad (34.73)$$

where $H_{L,n}$ is the negative log-likelihood Hessian with respect to the averaged logits. Because the averaged Jacobian enters quadratically, this curvature retains cross-transformation contributions. Current transformation samples provide a stochastic estimate that remains differentiable with respect to θ ; the prior curvature is added below.

Kronecker factorisation

KFAC makes the sampled GGN scalable by using a block-diagonal approximation across layers and Kronecker factors within each block [1136, 1402]. For a fully connected layer, $J_{n,s}^{(\ell)} = a_{n,s}^{(\ell-1)\top} \otimes B_{n,s}^{(\ell)}$, where $a_{n,s}^{(\ell-1)}$ are the bias-augmented input activations and $B_{n,s}^{(\ell)}$ is the output-to-preactivation Jacobian. We approximate

$$\bar{J}_n^{(\ell)} \approx \bar{a}_n^{(\ell-1)\top} \otimes \bar{B}_n^{(\ell)}, \quad (34.74)$$

where bars denote averages over the S transformation samples. This replaces an average of Kronecker products by a Kronecker product of averages and therefore discards activation–Jacobian correlations across samples. Applying the usual KFAC factorisation across data points gives

$$\hat{H}_{\text{GGN},S}^{(\ell)} = \sum_{n=1}^N \bar{J}_n^{(\ell)\top} H_{L,n} \bar{J}_n^{(\ell)} \approx N A^{(\ell)} \otimes G^{(\ell)}, \quad (34.75)$$

with

$$A^{(\ell)} = \frac{1}{N} \sum_{n=1}^N \bar{a}_n^{(\ell-1)} \bar{a}_n^{(\ell-1)\top}, \quad G^{(\ell)} = \frac{1}{N} \sum_{n=1}^N \bar{B}_n^{(\ell)\top} H_{L,n} \bar{B}_n^{(\ell)}. \quad (34.76)$$

For convolutional layers, KFAC additionally factorises over spatial sites.

34.8.3 Practical differentiable Laplace objective

We now combine this curvature approximation with a separate Gaussian prior precision $\lambda_\ell > 0$ for each layer. Let $P_\ell = \dim(w_\ell)$ and $p(w_\ell | \lambda_\ell) = \mathcal{N}(w_\ell | 0, \lambda_\ell^{-1} I)$. Its normalised log density is

$$\log p(w_\ell | \lambda_\ell) = \frac{P_\ell}{2} \log \lambda_\ell - \frac{\lambda_\ell}{2} \|w_\ell\|^2 - \frac{P_\ell}{2} \log(2\pi). \quad (34.77)$$

The prior curvature adds $\lambda_\ell I$ to the sampled KFAC block:

$$\hat{H}_{\text{KFAC},S}^{(\ell)} = N A^{(\ell)} \otimes G^{(\ell)} + \lambda_\ell I_{P_\ell}. \quad (34.78)$$

If $a_i^{(\ell)}$ and $g_j^{(\ell)}$ are the eigenvalues of $A^{(\ell)}$ and $G^{(\ell)}$, the block-diagonal log determinant is

$$\log |\hat{H}_{\text{KFAC},S}| = \sum_{\ell=1}^{L_{\text{layers}}} \sum_{i=1}^{D_{\ell}} \sum_{j=1}^{K_{\ell}} \log \left(N a_i^{(\ell)} g_j^{(\ell)} + \lambda_{\ell} \right), \quad (34.79)$$

where L_{layers} is the number of layers, $D_{\ell} = \dim(\bar{a}_n^{(\ell-1)})$, and K_{ℓ} is the preactivation dimension, so that $P_{\ell} = D_{\ell} K_{\ell}$ when biases are represented by augmented activations.

Substituting this log determinant into the Laplace expression yields

$$\log \hat{Z}_S^{\text{KFAC}}(\theta, \lambda) = \ell_S(\hat{w}_S, \theta) + \log p(\hat{w}_S | \lambda) - \frac{1}{2} \log |\hat{H}_{\text{KFAC},S}| + \frac{P}{2} \log(2\pi). \quad (34.80)$$

This is the differentiable Laplace (*Diff. Laplace*) objective optimised in practice. The factors $A^{(\ell)}$ and $G^{(\ell)}$ depend on θ through the sampled transformations, while the prior terms depend directly on λ . Training alternates between maximising $\ell_S(w, \theta) + \log p(w | \lambda)$ over w and updating θ and λ through the objective above.

The log determinant does not decompose over data points, so naively differentiating it would require retaining the full-data computation graph. For any positive-definite curvature block $H(\theta)$,

$$\nabla_{\theta} \log |H(\theta)| = \text{vec}(H(\theta)^{-1})^{\top} \nabla_{\theta} \text{vec}(H(\theta)). \quad (34.81)$$

This separates a preconditioner from a vector–Jacobian product. In practice, the full-data KFAC factors are first accumulated batch by batch without retaining a computation graph; their eigendecompositions are then computed and held fixed as the preconditioner. The remaining derivatives of $A^{(\ell)}$ and $G^{(\ell)}$ are sums over data points, so, conditional on this preconditioner, they admit unbiased minibatch estimates [815]. This gives a memory-efficient stochastic gradient of the KFAC–Laplace surrogate. During each hyperparameter update, $\hat{w}_S$ is also treated as fixed; accounting for its implicit dependence on (θ, λ) would require implicit differentiation or unrolling.

34.8.4 Experiments: learned input invariance

We next test whether Diff. Laplace recovers known transformations in controlled benchmarks and remains practical for larger convolutional networks [815].

Recovering dataset transformations

We train on rotated, translated, and affine-transformed variants of MNIST and monitor the learned invariance parameters θ . In Figure 34.9, parameters associated with the transformations used to generate each dataset increase toward their target ranges, while irrelevant candidates remain near zero.

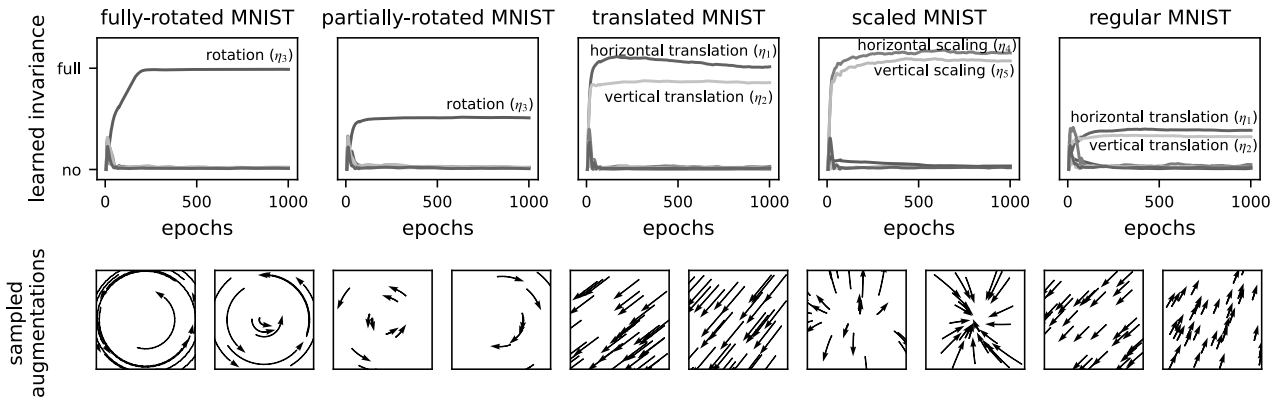


Figure 34.9: Trajectories of affine-invariance parameters $\theta = (\theta_1, \dots, \theta_6)$, corresponding to horizontal translation, vertical translation, rotation, horizontal scaling, vertical scaling, and shear, on transformed MNIST variants. Two samples from each learned transformation distribution are shown as flow fields. Adapted from Immer et al. [815].

Dataset	Network	Method	Fully Rotated	Partially Rotated	Translated	Scaled	Original
CIFAR-10	ResNet	non-invariant	54.16 ± 0.40	59.90 ± 0.12	69.65 ± 0.16	66.06 ± 0.13	74.13 ± 0.51
		Augerino	75.40 ± 0.19	74.76 ± 0.34	73.71 ± 0.31	72.07 ± 0.09	79.03 ± 1.04
		Diff. Laplace (ours)	79.50 ± 0.62	77.71 ± 0.46	79.21 ± 0.17	76.03 ± 0.15	84.19 ± 0.76

Table 34.2: Test accuracy (%) of ResNet models on transformed CIFAR-10 variants, reported as mean $\pm$ standard error over three seeds. Adapted from Immer et al. [815].

Method	Test accuracy (%)
non-invariant	85.17 ± 0.39
Augerino	87.67 ± 0.08
Diff. Laplace (ours)	91.98 ± 0.04

Table 34.3: CIFAR-10 test accuracy (%) for Wide ResNet input-invariance learning with KFAC. Adapted from Immer et al. [815].

Large ResNet models

Table 34.2 compares Diff. Laplace with a non-invariant ResNet and Augerino across five CIFAR-10 variants. Diff. Laplace obtains the highest mean test accuracy in every column, providing evidence that the method can learn useful input-level invariances in this architecture.

The Wide ResNet result in Table 34.3 provides a second scalability example using KFAC curvature. Here the transformation distribution is shared by the network as a whole and influences all layers through training; the next section instead assigns symmetry-controlling priors to individual layers.

Together, these experiments show that the same approximate model-selection objective can learn input-level invariances in both shallow and deep networks, including the ResNet architectures considered here.

34.8.5 Empirical comparison with explicit regularisation

We now compare Diff. Laplace with explicit regularisation on partially rotated MNIST. A representative method is Augerino [120], which optimises prediction loss under a learnable augmentation distribution and favours broader transformation ranges through

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{CE}} + \beta R(\theta), \quad (34.82)$$

where $\mathcal{L}_{\text{CE}}$ is the summed cross-entropy, $R(\theta)$ decreases as the encouraged transformation ranges broaden, and β controls the trade-off. If mean rather than summed cross-entropy is used, its factor of $1/N$ must be absorbed into β . Both approaches require a candidate transformation family, but explicit regularisation additionally requires the design of R and the choice of β .

Figure 34.10 shows that the invariance learned with explicit regularisation is sensitive to β , with some settings producing unstable runs. Diff. Laplace instead learns the target invariance without a separate regularisation coefficient.

The second comparison tests sensitivity to parameterisation. Let $\boldsymbol{\eta} = (\eta_1, \dots, \eta_6) \in \mathbb{R}_{>0}^6$ denote the same affine-range parameters previously represented by θ , and define the reciprocal coordinates $\zeta_i = 1/\eta_i$ componentwise. These coordinate systems describe the same strictly positive transformation ranges, with zero-range boundaries approached as limits.

Figure 34.11 shows that the explicit regulariser behaves differently after this coordinate change. By contrast, the exact marginal likelihood is unchanged by a one-to-one reparameterisation that preserves the induced probabilistic model. These experiments expose the two practical drawbacks of explicit regularisation discussed earlier: sensitivity to both the regularisation strength and the chosen coordinates. Diff. Laplace avoids a separate penalty, although its numerical approximation and optimisation can still exhibit coordinate dependence.

34.9 Learning layer-wise equivariances in deep neural networks

The previous sections learned input-level *invariance* by averaging predictions over transformations. Intermediate representations can instead be made *equivariant*: transforming the input then produces a corresponding transformation of the features. Such constraints can improve data efficiency and generalisation [210, 318, 914]. This raises a natural question: can we also learn *equivariances* in a principled, data-driven way?

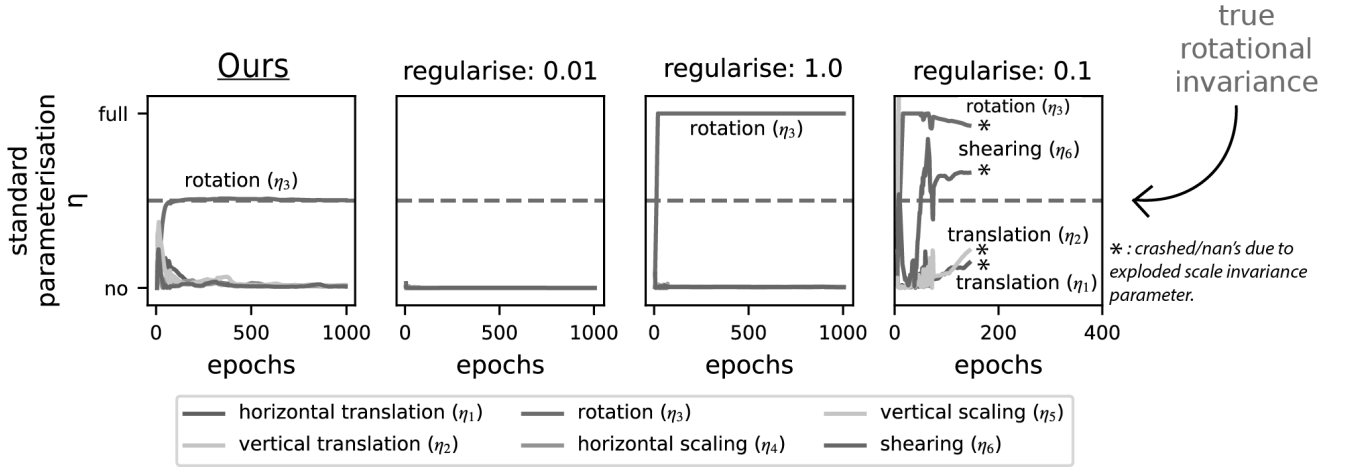


Figure 34.10: Learned affine-invariance parameters on partially rotated MNIST for Diff. Laplace and three explicit-regularisation strengths β . The dashed line marks the target rotational invariance; asterisks mark unstable runs. Adapted from Immer et al. [815].

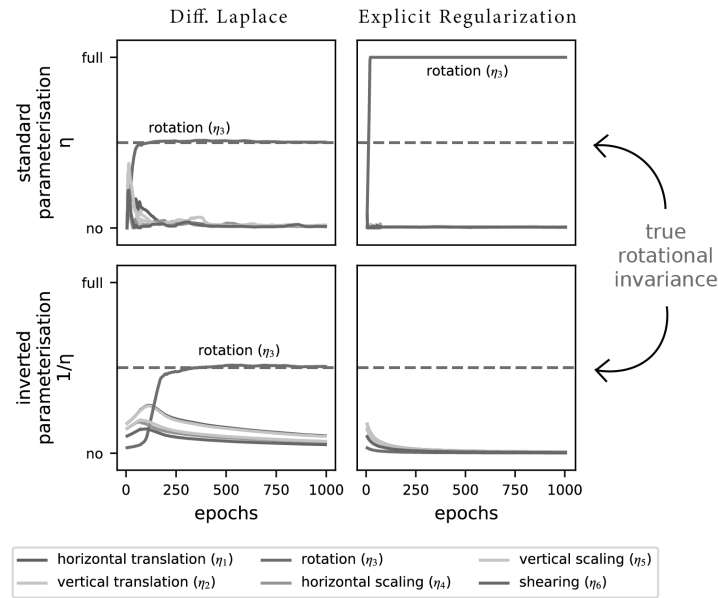


Figure 34.11: Learned affine-invariance parameters on partially rotated MNIST under the standard coordinates η (top) and componentwise reciprocal coordinates $\zeta = 1/\eta$ (bottom). Diff. Laplace is shown on the left and explicit regularisation on the right; dashed lines mark the target rotational invariance. Adapted from Immer et al. [815].

To make the degree of equivariance learnable, the model family must contain both equivariant and non-equivariant functions. We first review exact equivariance and two ways to relax it, then use Diff. Laplace to learn the relative prior strength of equivariant and non-equivariant pathways in each layer.

34.9.1 Exact equivariance and group convolution

Let $f(x) = Wx$, with $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, be a linear layer. Suppose $g \in G$ acts on the input and output spaces through T_{in}^g and T_{out}^g , respectively. Equivariance means that f intertwines these actions [319]:

$$f(T_{\text{in}}^g x) = T_{\text{out}}^g f(x) \quad \forall x \quad \Longleftrightarrow \quad WT_{\text{in}}^g = T_{\text{out}}^g W, \quad \forall g \in G. \quad (34.83)$$

If the same representation acts on both spaces, this becomes the commutation constraint

$$WT^g = T^g W, \quad \forall g \in G, \quad (34.84)$$

which defines a linear subspace of equivariant weight matrices.

Example: translations in 2D

Consider a feature field x indexed by $u \in \mathbb{Z}^2$, with channel indices suppressed. A translation by $t \in \mathbb{Z}^2$ acts as

$$(T^t x)[u] = x[u - t]. \quad (34.85)$$

On a finite periodic grid, the commutation constraint (34.84) requires the weights to depend only on relative displacement [319]:

$$W[u, u'] = \kappa[u' - u], \quad (Wx)[u] = \sum_{u'} \kappa[u' - u]x[u']. \quad (34.86)$$

Thus, convolution is the most general linear translation-equivariant map under these assumptions. Equivariance enforces weight sharing but not locality; ordinary convolutional networks add locality by restricting κ to finite support.

Equivariance to general groups

Let the input and output feature fields be indexed by elements of G and carry the left regular action

$$(T^h x)(g) = x(h^{-1}g), \quad h, g \in G. \quad (34.87)$$

For finite G , every equivariant linear map between such fields is a *group convolution*. In a notation that also covers locally compact continuous groups,

$$(Wx)(g) = \int_G \kappa(g^{-1}g')x(g') \, d\mu(g'). \quad (34.88)$$

Here μ is a left Haar measure; for finite or countable groups, the integral is interpreted as a sum with counting measure. The kernel depends only on the relative group element $g^{-1}g'$, and left invariance of μ makes the operator equivariant. For continuous groups, the kernel and features are assumed sufficiently integrable for the operator to be well defined. With multiple channels, κ is matrix-valued.

Example: roto-translations in 2D

To include rotations, consider

$$G = \mathbb{R}^2 \rtimes C_N, \quad (34.89)$$

where C_N is a finite cyclic rotation group. An image $x_0(u)$ is initially indexed only by position. A lifting layer correlates it with rotated filters to produce a group-indexed feature field $x(u, r)$, where $u \in \mathbb{R}^2$ and $r \in C_N$. A group element $(t, \rho) \in G$ then acts as

$$(T^{(t, \rho)} x)(u, r) = x(R_\rho^{-1}(u - t), \rho^{-1}r). \quad (34.90)$$

Here, R_ρ is the planar rotation associated with ρ .

Replacing C_N by continuous rotations gives $\text{SE}(2) = \mathbb{R}^2 \rtimes \text{SO}(2)$. On a pixel grid, either action requires discretisation and, for off-grid coordinates, interpolation. The resulting numerical equivariance is therefore approximate.

Substituting this action into the general group-convolution form gives

$$(Wx)(u, r) = \sum_{r' \in C_N} \int_{\mathbb{R}^2} \kappa(R_r^{-1}(u' - u), r^{-1}r') x(u', r') du'. \quad (34.91)$$

The kernel depends only on displacement in the output orientation frame and on relative orientation. It therefore shares weights jointly across positions and orientations, giving exact equivariance in the continuous-domain model and approximate equivariance after discretisation.

34.9.2 Relaxation through nonstationary kernels

Strict group convolutions depend only on the relative element $g^{-1}g'$. To relax equivariance, we additionally allow the kernel to depend on the absolute output coordinate g [1659]. Suppressing channel indices, a general kernel operator between group-indexed features is

$$(Wx)(g) = \int_G k(g, g') x(g') d\mu(g'). \quad (34.92)$$

For finite or countable groups, the integral is replaced by a sum. Since the change from (g, g') to the relative-absolute coordinates $(g^{-1}g', g)$ is one-to-one, we may write

$$k(g, g') = \tilde{k}(g^{-1}g', g), \quad (34.93)$$

$$(Wx)(g) = \int_G \tilde{k}(g^{-1}g', g) x(g') d\mu(g'). \quad (34.94)$$

If $\tilde{k}$ is independent of its second argument, this reduces to an exactly equivariant group convolution. Dependence on g permits symmetry-breaking variation across absolute output coordinates. A practical parameterisation combines relative and absolute embeddings through a mixer:

$$\tilde{k}(g^{-1}g', g) = m(\psi_{\text{rel}}(g^{-1}g'), \psi_{\text{abs}}(g)), \quad (34.95)$$

where ψ_{rel} and ψ_{abs} are learned embeddings and m is a mixing function. Exact equivariance is recovered when m ignores its second argument. The expressivity of the resulting nonstationary kernel depends on the embeddings and mixer.

Smooth relative and absolute features

Random Fourier features provide a smooth parameterisation of the relative and absolute embeddings [1363]. Let $\nu_r \sim \mathcal{N}(0, I_d)$, $r = 1, \dots, R$, be fixed random projection directions and let $\xi : G \rightarrow \mathbb{R}^d$ be a coordinate embedding. For a frequency scale ω , define

$$\varphi_\omega(g) = \frac{1}{\sqrt{R}} [\cos(\omega \nu_1^\top \xi(g)), \sin(\omega \nu_1^\top \xi(g)), \dots, \cos(\omega \nu_R^\top \xi(g)), \sin(\omega \nu_R^\top \xi(g))]^\top. \quad (34.96)$$

Convenient coordinate maps for translations, rotations, and roto-translations are

$$\xi(t) = t, \quad \xi(\rho) = [\cos \rho, \sin \rho]^\top, \quad (34.97)$$

$$\xi(t, \rho) = [t_1, t_2, \cos \rho, \sin \rho]^\top. \quad (34.98)$$

The periodic angle coordinates ensure that the features are well defined on $\text{SO}(2)$. Other groups may require group-specific coordinate maps or bases [318, 520, 914].

Suppressing input-output channel indices, write

$$\psi_{\text{rel}}(g^{-1}g') = \mathbf{w}_{\text{rel}}^\top \varphi_{\omega_{\text{rel}}}(g^{-1}g'), \quad (34.99)$$

$$\psi_{\text{abs}}(g) = \mathbf{w}_{\text{abs}}^\top \varphi_{\omega_{\text{abs}}}(g). \quad (34.100)$$

The random directions are shared across channels, while separate coefficients $\mathbf{w}_{c_{\text{out}},c_{\text{in}}}^{(\text{rel})}$ and $\mathbf{w}_{c_{\text{out}},c_{\text{in}}}^{(\text{abs})}$ are learned for each channel pair.

The two scales have different roles. Changing ω_{rel} changes the smoothness of the filter as a function of $g^{-1}g'$, while preserving equivariance. The absolute features can instead break equivariance: when $\omega_{\text{abs}} = 0$, $\varphi_{\omega_{\text{abs}}}(g)$ is constant in g , whereas nonzero values allow the kernel to vary across absolute output coordinates. The realised amount of symmetry breaking also depends on the absolute coefficients and the mixer m ; exact equivariance is recovered whenever the resulting kernel is independent of its absolute argument.

Additive implementation and its limitation

For computational efficiency, the mixer can be replaced by the additive special case $\tilde{k}(g^{-1}g', g) = \psi_{\text{rel}}(g^{-1}g') + \psi_{\text{abs}}(g)$. For input feature $x_{c_{\text{in}}}(g')$ and output feature $y_{c_{\text{out}}}(g)$, this gives

$$y_{c_{\text{out}}}(g) = \sum_{c_{\text{in}}=1}^{C_{\text{in}}} \int_G \left[\mathbf{w}_{c_{\text{out}},c_{\text{in}}}^{(\text{rel})\top} \varphi_{\omega_{\text{rel}}}(g^{-1}g') + \mathbf{w}_{c_{\text{out}},c_{\text{in}}}^{(\text{abs})\top} \varphi_{\omega_{\text{abs}}}(g) \right] x_{c_{\text{in}}}(g') d\mu(g'). \quad (34.101)$$

The relative term is equivariant, while the absolute term can break the symmetry. Because its basis does not depend on the input element g' , the absolute contribution factorises as

$$\sum_{c_{\text{in}}=1}^{C_{\text{in}}} b_{c_{\text{out}},c_{\text{in}}}(g) \int_G x_{c_{\text{in}}}(g') d\mu(g'). \quad (34.102)$$

Here $b_{c_{\text{out}},c_{\text{in}}}(g) := \mathbf{w}_{c_{\text{out}},c_{\text{in}}}^{(\text{abs})\top} \varphi_{\omega_{\text{abs}}}(g)$. This is a low-rank global correction: it can break equivariance, but it cannot represent a general nonstationary map. Full flexibility requires joint relative–absolute features, for example a tensor-product basis depending on both $g^{-1}g'$ and g . Figure 34.12 illustrates the broader family of joint nonstationary translation kernels; the additive construction above represents only a restricted subset.

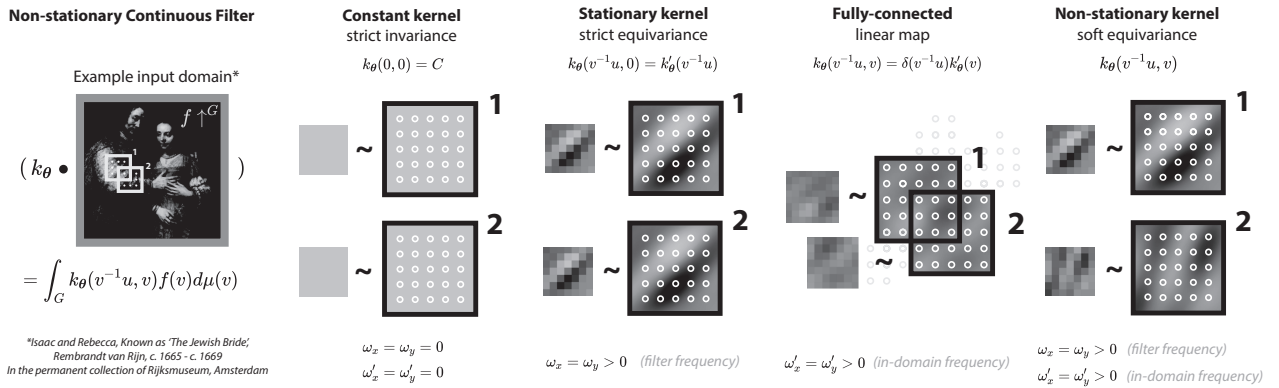


Figure 34.12: Translation kernels obtained by varying the relative (filter) and absolute-coordinate (output-domain, in our convention) frequencies. For the general joint nonstationary kernel, the limiting cases include a constant invariant kernel, a stationary equivariant kernel, a fully connected linear map, and a softly equivariant kernel. The additive parameterisation above is a restricted special case. Adapted from van der Ouderaa et al. [1659].

34.9.3 Relaxation through residual pathways

Nonstationary kernels relax equivariance through absolute-coordinate dependence. An alternative—and the one used in the experiments below—is to add a non-equivariant residual pathway to an equivariant one [521]:

$$y^{(\ell)} = W_{\text{eq}}^{(\ell)} x^{(\ell-1)} + W_{\text{non}}^{(\ell)} x^{(\ell-1)}. \quad (34.103)$$

For a finite group, $W_{\text{eq}}^{(\ell)}$ has a group-convolutional kernel $\mathbf{w}_{\text{eq}}[g^{-1}g']$, while $W_{\text{non}}^{(\ell)}$ may use an unrestricted kernel $\mathbf{w}_{\text{non}}[g, g']$. For translations, these are convolutional and fully connected pathways, respectively.

Independent zero-mean Gaussian priors on the two parameter blocks define a *residual pathway prior*. Let $\lambda_{\ell, \text{non}}$ denote the precision of the non-equivariant parameters. Increasing this precision suppresses the residual

pathway, with exact equivariance recovered in the infinite-precision limit. The original method fixes or tunes the relative prior variances and trains with a MAP objective; here, Diff. Laplace instead learns the layer-wise precisions through approximate marginal likelihood.

An unrestricted residual pathway has $\mathcal{O}(C_{\text{in}}C_{\text{out}}|G|^2)$ parameters, or $\mathcal{O}(D^4)$ on a $D \times D$ grid. The experiments therefore use factorised or sparsified variants that retain non-equivariant flexibility with far fewer parameters [1658]. Other approaches also relax equivariance [1456, 1600, 1657, 1710], but the residual decomposition is particularly convenient here because separate priors can directly control the two pathways.

34.9.4 Experiments: learned layer-wise equivariance

Using these scalable residual pathways, we ask whether marginal-likelihood optimisation can mitigate symmetry misspecification, select an architecture from training data that generalises well, reveal where layer-wise symmetry constraints are retained, and choose among candidate equivariant groups.

Mitigating symmetry misspecification

To isolate symmetry misspecification, digits are placed in different image quadrants. In the first task, the target is digit identity and is invariant to position. In the second, the target contains both the digit and its quadrant, so enforcing position invariance discards information required for prediction. Conversely, a fully connected model retains position but fails to exploit the genuine symmetry of the first task. Table 34.4 compares these fixed alternatives with a model whose pathway precisions are selected by Diff. Laplace.

Example:  (3, upper left)			MAP		Diff. Laplace		
			FC	CONV	FC	CONV	Learned
	Task symmetry	Prediction target	112.7 M	0.4 M	112.7 M	0.4 M	1.8 M
	Strict symmetry	<i>digit</i>	95.73	99.38	97.04	99.23	98.86
	Partial symmetry	<i>digit, quadrant</i>	95.10	24.68	95.46	24.50	99.00

Table 34.4: Test accuracy (%) of fully connected (FC), convolutional (CONV) with an invariant readout, and learned-equivariance models on digit-only and digit-and-quadrant prediction tasks. Parameter counts appear below the model names. Adapted from van der Ouderaa et al. [1658].

The convolutional model performs well when the target is position invariant but falls to near-chance accuracy when the quadrant must be predicted. The fully connected model avoids this misspecification but does not exploit the symmetry as effectively. By adjusting the non-equivariant pathway precision, Diff. Laplace achieves high accuracy on both tasks.

Selecting a generalising architecture from training data

On CIFAR-10, we compare non-equivariant, strictly convolutional, and adaptive FC+CONV architectures under MAP and Diff. Laplace training. The pathway precisions are selected using only the training data; the test metrics in Table 34.5 are used afterward to assess how well the selected architecture generalises.

Layer	Learnable equivariance	# Params	MAP		Diff. Laplace	
			Test NLL (↓)	Test Acc. (↑)	Test NLL (↓)	Test Acc. (↑)
Non-equivariant (FC path)		0.8 M	5.614	50.21	1.496	46.18
Strict equivariance (CONV path)		0.2 M	0.604	82.53	0.562	81.29
Learned equivariance (FC+CONV paths)	x	1.0 M	5.485	58.80	0.538	81.55

Table 34.5: CIFAR-10 test negative log likelihood (NLL) and accuracy (%) for non-equivariant, strictly equivariant, and learned-equivariance models under MAP and Diff. Laplace. Adapted from van der Ouderaa et al. [1658].

Under Diff. Laplace, the adaptive model improves test NLL from 0.562 to 0.538 and accuracy from 81.29% to 81.55% relative to strict equivariance. The same FC+CONV parameterisation performs substantially worse under MAP. Thus, its success comes from the marginal-likelihood objective selecting an effective architecture from the training data, rather than merely from making both pathways available.

Inspecting the learned layer-wise architecture

We next inspect which symmetry constraints the marginal likelihood selected in each layer. The learned non-equivariant-path precisions $\lambda_{\ell, \text{non}}$ exceed 10^6 in most layers, indicating strong shrinkage. Figure 34.13 visualises the resulting architecture using relative effective parameter counts, a curvature-based diagnostic of how much each pathway remains active after prior shrinkage [1105]. This diagnostic is not itself a direct measure of functional contribution. Convolutional pathways dominate until the spatial resolution reaches 1×1 ; at that resolution, an FC pathway is equivalent to a pointwise convolution and does not necessarily indicate broken spatial equivariance.

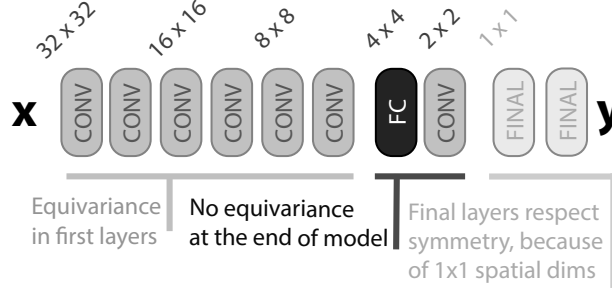


Figure 34.13: Dominant pathway across the depth of a learned FC+CONV network, determined from relative effective parameter counts. Convolutional paths dominate until the spatial resolution reaches 1×1 . Adapted from van der Ouderaa et al. [1658].

Selecting among candidate symmetry groups

The preceding experiments supplied one equivariant pathway and one unrestricted alternative. More generally, each layer can contain several candidates with separately learned prior strengths. Here, FC is unrestricted, CONV is translation equivariant, and GCONV is a group-convolutional pathway equivariant to translations and the discrete rotations C_N , corresponding to $\mathbb{R}^2 \rtimes C_N$. Table 34.6 reports predictive performance, approximate evidence, and the relative effective parameter counts used to inspect which pathways remain active.

Dataset	# Params	MAP		Learned with Diff. Laplace			Relative effective parameter count			
		Test NLL ($\downarrow$)	Test accuracy (%)	Test NLL ($\downarrow$)	Test accuracy (%)	$-\log \widehat{p(\mathcal{D})}/N$ ($\downarrow$)	FC (%)	CONV (%)	GCONV (%)	
MNIST	1.2 M	0.172	97.59	0.023	99.21	0.328	10 (0-46)	15 (0-98)	75 (1-100)	
Translated MNIST	1.2 M	0.812	90.78	0.053	98.27	0.216	0 (0-0)	23 (0-99)	77 (1-100)	
Rotated MNIST	1.2 M	0.819	91.02	0.136	95.55	0.896	8 (0-20)	8 (0-47)	83 (47-100)	
CIFAR-10	1.2 M	3.540	68.33	0.552	80.94	0.926	0 (0-1)	44 (0-99)	56 (0-100)	
Rotated CIFAR-10	1.2 M	5.953	48.30	1.236	55.68	1.630	4 (0-22)	14 (0-41)	82 (58-99)	

Table 34.6: Selection among fully connected (FC), convolutional (CONV), and rotation-equivariant group-convolutional (GCONV) pathways. In $-\log \widehat{p(\mathcal{D})}/N$, the hat denotes the Laplace approximation and division by N normalises the value per datum; the final columns give the mean (minimum–maximum) relative effective parameter count across layers. Adapted from van der Ouderaa et al. [1658].

The small FC effective counts indicate that non-equivariant flexibility is strongly suppressed, particularly on translated MNIST and CIFAR-10. GCONV receives the largest allocation throughout and is most dominant on the rotated datasets, while CONV remains active on several tasks. These diagnostics are consistent with the method selecting among the supplied pathways and adjusting their strengths by layer; they do not imply discovery of groups outside the candidate architecture.

Together, the experiments show that marginal-likelihood optimisation can relax a misspecified symmetry, select a generalising architecture using only training data, and reveal where different symmetry constraints remain active within that architecture.

34.10 Conclusion and Outlook

In this chapter, we have argued that without Bayesian model selection, Bayesian deep learning is missing out on one of its key abilities: the ability to improve generalisation by selecting an appropriate inductive bias. We have reviewed the Bayesian approach to this problem, and argued that the challenges in making this work are not with

the conceptual Bayesian framework, but instead with finding usable approximations to the prescribed quantities. We could make progress towards creating adaptive neural network architectures, even with crude approximations to the marginal likelihood.

Whether adaptive architectures can provide benefits at the scale of modern foundation models, and with modern architectures like transformers, remains an open question. Perhaps model size is the next important question to tackle [1122, 1189] and perhaps Bayesian approaches may yet again have something to offer [1315].

Chapter 35

Interpretable and identifiable deep generative models via Bayesian pyramids and their variants

Deep generative models (DGMs) have become central tools for representation learning and achieved great empirical success across many applications. However, the theoretical foundations for parameter and representation identifiability, which is a prerequisite for interpretability and reliable Bayesian inference, still remain limited. This chapter develops a unified perspective on *identifiable and interpretable deep generative models with discrete latent layers*, synthesising several recently proposed models. These frameworks establish explicit structural and algebraic conditions under which the multi-layer latent structure and its parameters are uniquely determined from the observable distribution. Such identifiability guarantees provide solid foundations for reliably applying these models in high-stakes applications, such as education, medicine, and health care. We outline the theoretical results, sketch intuition via algebraic and tensor-based reasoning, and discuss implications for interpretable Bayesian deep learning.

35.1 Introduction

DGMs such as deep belief networks, variational autoencoders, and generative adversarial networks have achieved striking empirical performance in modelling complex data. Yet, from a statistical perspective, most existing DGMs are *nonidentifiable*: distinct parameter values can induce the same distribution of observables. This ambiguity impedes interpretability, obstructs posterior consistency, and causes label-switching and other pathologies in Bayesian inference.

Identifiability roughly means that unknown quantities, such as parameters and latent structures, can be uniquely identified from the observed data. It is a crucial prerequisite for reliable scientific discoveries and reproducible statistical analyses. Although classical latent variable models such as factor analysis and latent class models have a well-established identifiability theory, their deep counterparts remain poorly understood.

This chapter presents several recently proposed identifiable DGMs based on discrete latent structures: the *Bayesian Pyramid* model for multivariate categorical data, Bayesian latent class regression with interpretable binary profiles, Bayesian deep generative models for multiplex networks, and *Deep Discrete Encoders* with many discrete latent layers for rich data types coming from flexible distributions. Together, these models demonstrate that Bayesian deep learning with many random latent layers can be both *interpretable and identifiable* under explicit structural conditions.

35.2 Bayesian Pyramids

Multivariate categorical data are routinely collected in biomedical and social sciences. It is of great importance to build interpretable models that perform dimension reduction and uncover meaningful latent structures from such discrete data. Identifiability is a fundamental requirement for interpretable modelling and inference in such scenarios, yet is challenging to address when there are complex latent structures. *Bayesian Pyramids* proposed by Gu and Dunson [679] are a class of identifiable multilayer/deep discrete latent structure models for multivariate

categorical data.

35.2.1 Model Structure and Identifiability Results

The Bayesian Pyramid is a directed graphical model for multivariate categorical data. In a Bayesian Pyramid, the bottom layer is the observed data vector $\mathbf{y} \in [d_1] \times [d_2] \times \cdots \times [d_p]$, and many binary latent variables are organised in multiple layers above the observed layer, with the shallowest layer being $\boldsymbol{\alpha}^{(1)}$ and the deepest layer being $\boldsymbol{\alpha}^{(D)}$. Directed arrows occur only between adjacent layers in the top-down direction: $\boldsymbol{\alpha}^{(D)} \rightarrow \boldsymbol{\alpha}^{(D-1)} \rightarrow \cdots \rightarrow \boldsymbol{\alpha}^{(1)} \rightarrow \mathbf{y}$. For a random variable X , define $\text{Pa}(X)$ to be the index set of the variables with directed arrows pointing to X (called the *parents* of X). Each layer $\boldsymbol{\alpha}^{(\ell)} \in \{0, 1\}^{K_\ell}$ represents binary latent attributes. Formally, for any subject i , the joint distribution of all random variables factorises according to the graphical model as follows:

$$\mathbb{P}(\mathbf{y}_i, \{\boldsymbol{\alpha}_i^{(m)}\}, z_i^{(D)}) = \mathbb{P}(\mathbf{y}_i \mid \boldsymbol{\alpha}_i^{(1)}) \prod_{m=1}^{D-2} \mathbb{P}(\boldsymbol{\alpha}_i^{(m)} \mid \boldsymbol{\alpha}_i^{(m+1)}) \mathbb{P}(\boldsymbol{\alpha}_i^{(D-1)} \mid z_i^{(D)}) \mathbb{P}(z_i^{(D)}); \quad (35.1)$$

$$\mathbb{P}(\mathbf{y}_i \mid \boldsymbol{\alpha}_i^{(1)}) = \prod_{j=1}^p \mathbb{P}(y_{i,j} \mid \boldsymbol{\alpha}_{i, \text{Pa}(j)}^{(1)}), \quad \mathbb{P}(\boldsymbol{\alpha}_i^{(m)} \mid \boldsymbol{\alpha}_i^{(m+1)}) = \prod_{k=1}^{K_m} \mathbb{P}(\alpha_{i,k}^{(m)} \mid \boldsymbol{\alpha}_{i, \text{Pa}(k^{(m)})}^{(m+1)}). \quad (35.2)$$

Importantly, each variable (either in the observed layer or in any latent layer) connects only to a small subset of latent nodes above it. To summarise such parent–child dependence relationships, we introduce *graphical matrices* $\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(D-1)}$. Here, $\mathbf{G}^{(1)} = (g_{jk}^{(1)}) \in \{0, 1\}^{p \times K_1}$ encodes the bipartite graph between the $\mathbf{y}$ layer and the $\boldsymbol{\alpha}^{(1)}$ layer, where $g_{jk}^{(1)} = 1$ if y_j is a child of $\alpha_k^{(1)}$ and 0 otherwise. Similarly, for each $m = 2, \dots, D-1$, the matrix $\mathbf{G}^{(m)} = (g_{k\ell}^{(m)}) \in \{0, 1\}^{K_{m-1} \times K_m}$ describes the bipartite graph between the layers $\boldsymbol{\alpha}^{(m-1)}$ and $\boldsymbol{\alpha}^{(m)}$, where $g_{k\ell}^{(m)} = 1$ if $\alpha_k^{(m-1)}$ is a child of $\alpha_\ell^{(m)}$ and 0 otherwise. Bayesian Pyramids are identifiable under the following transparent algebraic conditions on the shapes of these matrices $\{\mathbf{G}^{(m)}\}$.

Theorem 35.1 (Theorem 4 in Gu and Dunson [679]). *Consider a Bayesian Pyramid where the numbers $K_1, \dots, K_{D-1}$ are known. Suppose each binary graphical matrix $\mathbf{G}^{(m)}$ of size $K_{m-1} \times K_m$ (size $p \times K_1$ if $m = 1$) takes the following form after some row permutation,*

$$\mathbf{G}^{(m)} = \begin{pmatrix} \mathbf{I}_{K_m} \\ \mathbf{I}_{K_m} \\ \mathbf{I}_{K_m} \\ \mathbf{G}^{(m),*} \end{pmatrix}, \quad m = 1, \dots, D-1, \quad (35.3)$$

where $\mathbf{G}^{(m),*}$ denotes a submatrix of $\mathbf{G}^{(m)}$ with an arbitrary form. Then the following parameters are identifiable up to a latent variable permutation within each layer: the distribution of the deepest latent variable $z^{(D)}$, the conditional probability table of each variable (including observed and latent) given its parents, and also the graphical matrices $\{\mathbf{G}^{(m)}; m = 1, \dots, D-1\}$.

A natural consequence of identifiability is posterior consistency, presented in the next theorem.

Theorem 35.2 (Theorem 3 in Gu and Dunson [679]). *Denote the collection of model parameters by Θ . Suppose the prior distributions for the parameters all have full support around the true values. If the parameters satisfy the proposed strict identifiability conditions, then*

$$\mathbb{P}(\Theta \in \mathcal{N}_\epsilon^c(\Theta^0) \mid \mathbf{y}_1, \dots, \mathbf{y}_n) \rightarrow 0 \text{ almost surely,}$$

where $\mathcal{N}_\epsilon^c(\Theta^0)$ is the complement of an ϵ -neighbourhood of the true Θ^0 in the parameter space.

Theorem 35.2 implies that under an identifiable model and with appropriately specified priors, the posterior concentrates around the true parameters rather than a manifold of equivalent parameterisations. This property guarantees reproducible inference and interpretability of the latent structure – a feature absent in many neural network-based DGMs.

35.2.2 Bayesian Inference for Two-layer Bayesian Pyramids

Here $z_i \in [B]$ denotes the subject-specific deepest latent class, with $\mathbb{P}(z_i = b) = \tau_b$, and $\eta_{k,b} = \mathbb{P}(\alpha_{i,k}^{(1)} = 1 \mid z_i = b)$. In this section, as an important special case, we focus on a two-latent-layer model as illustrated in Figure 35.1

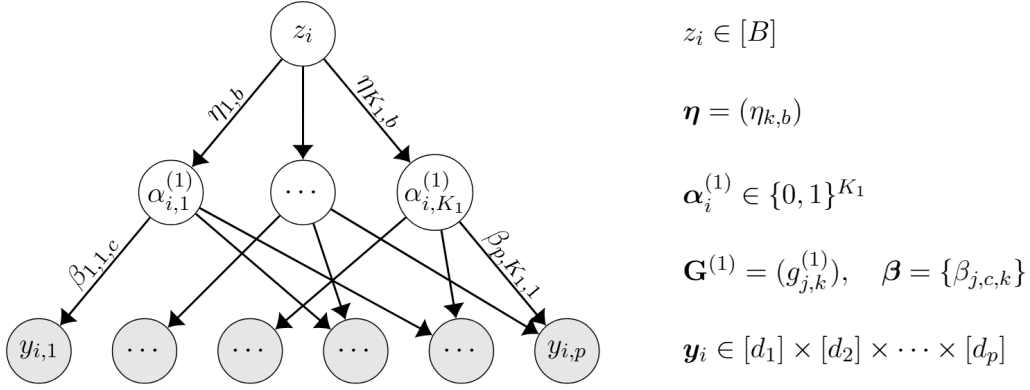


Figure 35.1: Two-latent-layer model. Latent variables $z_i, \alpha_{i,1}, \dots, \alpha_{i,K_1}$ and observed variables $y_{i,1}, \dots, y_{i,p}$ are subject-specific, and model parameters $\mathbf{G}^{(1)}, \beta, \tau, \eta$ are population quantities.

with depth $D = 2$ and use a Bayesian approach to infer the latent structure and model parameters. Assume

$$\begin{aligned} \mathbb{P}(y_{i,j} = c \mid \alpha_i^{(1)} = \alpha) &= \frac{\exp\left(\beta_{j,c,0} + \sum_{k=1}^{K_1} \beta_{j,c,k} g_{j,k}^{(1)} \alpha_k\right)}{\sum_{m=1}^{d_j} \exp\left(\beta_{j,m,0} + \sum_{k=1}^{K_1} \beta_{j,m,k} g_{j,k}^{(1)} \alpha_k\right)}, \quad j \in [p], \quad c \in [d_j]; \\ \mathbb{P}(\alpha_i^{(1)} = \alpha) &= \sum_{b=1}^B \tau_b \prod_{k=1}^{K_1} \eta_{k,b}^{\alpha_k} (1 - \eta_{k,b})^{1 - \alpha_k}, \quad \alpha \in \{0, 1\}^{K_1}. \end{aligned} \quad (35.4)$$

Here we assume $\beta_{j,d_j,0} \equiv \beta_{j,d_j,k} \equiv 0$ for all $k \in [K_1]$. The parameter $\beta_{j,c,k}$ can be viewed as the weight associated with the potential directed edge from latent variable $\alpha_k^{(1)}$ to observed variable y_j , for response category c . The $\beta_{j,c,k}$ only impacts the likelihood if there is an edge from $\alpha_k^{(1)}$ to y_j with $g_{j,k}^{(1)} = 1$. Denote the collection of β parameters as β . (35.4) specifies a usual latent class model in the second latent layer with B latent classes for the K_1 -dimensional latent vector $\alpha^{(1)}$. This layer of the model has latent class proportion parameters $\tau = (\tau_1, \dots, \tau_B)^\top$ and conditional probability parameters $\eta = (\eta_{k,b})_{K_1 \times B}$. The full set of model parameters across layers is $(\mathbf{G}^{(1)}, \beta, \tau, \eta)$.

In addition to the previous *strict identifiability* result in Theorem 35.1 that holds for all parameters in the parameter space, for model (35.4) we can also establish a slightly weaker notion of *generic identifiability* under much more relaxed conditions on $\mathbf{G}^{(1)}$. Roughly speaking, generic identifiability means that there exists a measure-zero subset in the parameter space in which non-identifiability may occur. This notion was popularised by Allman et al. [22] and usually suffices for real data analysis purposes. Before stating the identifiability result, we formally define the allowable constrained parameter space for β under a graphical matrix $\mathbf{G}^{(1)}$ as

$$\Omega(\beta; \mathbf{G}^{(1)}) = \{\beta_{1:p, 1:K_1, 1:(d_j-1)}; \beta_{j,c,k} \neq 0 \text{ if } g_{j,k}^{(1)} = 1; \text{ and } \beta_{j,c,k} = 0 \text{ if } g_{j,k}^{(1)} = 0\}. \quad (35.5)$$

Proposition 35.1 (Proposition 3 in Gu and Dunson [679]). *Consider model (35.4) with β belonging to $\Omega(\beta; \mathbf{G}^{(1)})$ in (35.5). Suppose $K_1 \geq 2\lceil \log_2 B \rceil + 1$ and the graphical matrix $\mathbf{G}^{(1)}$ can be rewritten as $\mathbf{G}^{(1)} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, (\mathbf{G}^*)^\top)^\top$, where each $\mathbf{G}_m$ has size $K_1 \times K_1$ and*

$$\mathbf{G}_m = \begin{pmatrix} 1 & * & \dots & * \\ * & 1 & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \dots & 1 \end{pmatrix}, \quad m = 1, 2, 3;$$

i.e., each of $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$ has all diagonal entries equal to one while any off-diagonal entry can be either one or zero. Then $(\mathbf{G}^{(1)}, \beta, \tau, \eta)$ are generically identifiable.

Bayesian Inference Procedure. We first describe the case where the true number of binary latent variables K_1 in the middle layer is fixed. We impose a Dirichlet prior on τ and Beta priors on entries of η . Inferring the latent sparse graph $\mathbf{G}^{(1)}$ is equivalent to inferring the sparsity structure of the continuous parameters β_{jkc} 's. Let

the prior for $\beta_{j,k,0}$ be $N(\mu_0, \sigma_0^2)$, where hyperparameters μ_0, σ_0^2 can be set to give weakly informative priors for the logistic regression coefficients. Specify priors for other parameters as

$$\begin{aligned}
 \text{(i) for fixed } K_1, k = 1, \dots, K_1 : \quad & \beta_{jck} \mid (\sigma_{ck}^2, g_{j,k} = 1) \sim N(0, \sigma_{ck}^2); \\
 & \beta_{jck} \mid (\sigma_{ck}^2, g_{j,k} = 0) \sim N(0, \sigma_0^2); \\
 & \sigma_{ck}^2 \sim \text{InvGa}(b_{1\sigma}, b_{2\sigma}); \\
 & \mathbb{P}(g_{j,k} = 1) = 1 - \mathbb{P}(g_{j,k} = 0) = \gamma;
 \end{aligned} \tag{35.6}$$

Here σ_0 is a small positive number specifying the “pseudo”-variance, and we take $\sigma_0 = 0.1$ in the numerical studies. Our adoption of a prior variance for $\beta_{j,c,k}$ when $g_{j,k} = 0$ follows a similar spirit as the “pseudo-prior” approach in the Bayesian variable selection literature.

Next, we describe the prior specifications with an unknown number of binary latent variables K_1 . Consider an upper bound K_{upper} for K_1 . Based on the identifiability conditions in Theorem 35.1 about the shape of $\mathbf{G}_{p \times K_1}^{(1)}$, K_1 is naturally constrained to be at most $p/3$, therefore K_{upper} can be set to $p/3$ or smaller in practice. We adopt a prior that detects redundant binary latent variables by increasingly shrinking the variance of β_{jck} ’s as k grows from 1 to K_{upper} . Specifically, letting $\beta_{jck} \sim N(0, \sigma_{ck}^2)$, we put a Cumulative Shrinkage Prior (CUSP) prior [1009] on the variances $\{\sigma_{c1}^2, \sigma_{c2}^2, \dots, \sigma_{cK_{\text{upper}}}^2\}$ for each category $c \in [d-1]$, where σ_∞^2 is a prespecified small positive number indicating the spike variance for redundant binary latent variables:

(ii) for unknown $K_1 < K_{\text{upper}}$, $k = 1, \dots, K_{\text{upper}}$:

$$\sigma_{ck}^2 \mid \pi_k \sim (1 - \pi_k) \text{InvGa}(b_{1\sigma}, b_{2\sigma}) + \pi_k \delta_{\sigma_\infty^2}; \tag{35.7}$$

$$\pi_k = \sum_{\ell=1}^k v_\ell \prod_{m=1}^{\ell-1} (1 - v_m), \tag{35.8}$$

where $\text{InvGa}(b_{1\sigma}, b_{2\sigma})$ refers to the inverse gamma distribution with shape $b_{1\sigma}$ and scale $b_{2\sigma}$, and π_k has a stick-breaking representation as in (35.8) with $v_1, v_2, \dots, v_{K_{\text{upper}}-1}$ independently following the Beta distribution $\text{Beta}(1, \alpha_0)$. We set $v_{K_{\text{upper}}} \equiv 1$ to truncate the stick-breaking representation at K_{upper} , similarly to Legramanti et al. [1009]. Posterior computation is performed with the data-augmented Gibbs sampler developed in Gu and Dunson [679].

35.2.3 Simulation Results and Real Data Application

Simulation Studies. We perform simulation studies to empirically verify the identifiability results and evaluate the performance of our algorithm. Consider a two-layer Bayesian Pyramid with $p = 20$ observed variables, $d = 4$ response categories for each observed variable, and $K_1 = 4$ binary latent variables in the middle layer and one binary latent variable in the deep layer. Let the true graphical matrix be $\mathbf{G}^{(1)} = (\mathbf{I}_4; \mathbf{I}_4; \mathbf{I}_4; 1100; 0110; 0011; 1001; 1010; 1001; 0101; 1010; 1001; 0101; 1010; 1001; 0101; 1010; 1001; 0101; 1010)$ which satisfies the strict identifiability condition in Theorem 35.1. Let the intercepts for categories $c = 1, 2, 3$ be $(\beta_{j,1,0}, \beta_{j,2,0}, \beta_{j,3,0}) = (-3, -2, -1)$ for each j ; and for any $g_{j,k}^{(1)} = 1$, let the corresponding true main-effect parameters be $\beta_{j,c,k} = 3$ if variable y_j has a single parent and $\beta_{j,c,k} = 2$ if y_j has multiple parents. See Figure 35.2(c) for a heatmap of the sparse matrix of the main-effect parameters $(\beta_{j1k}; j \in [p], k \in [K_1])$ for category $c = 1$.

We use the method with the CUSP prior and specify $K_{\text{upper}} = 7$, because $\lceil p/3 \rceil = 7$ is a natural upper bound for K_1 to satisfy the identifiability condition in Theorem 35.1. As for the hyperparameters in the CUSP prior, we mainly follow the default setting and suggestion in Legramanti et al. [1009] and set $\alpha_0 = 5$ and $\sigma_\infty^2 = 0.07$. First, Figure 35.2 presents posterior means of the main-effect parameters $(\beta_{j1k}; j \in [p], k \in [K_1])$ for category $c = 1$, averaged across simulation replications. This figure demonstrates that the number of latent variables K_1 can be correctly inferred from our algorithm. Furthermore, Figures 35.3 and 35.4 show that as sample size n grows, both the discrete $\mathbf{G}$ and the continuous parameters are estimated consistently. These results empirically confirm the posterior consistency in Theorem 35.2.

Application to the DNA Nucleotide Sequence Data. We apply the proposed method for two-layer Bayesian Pyramids and analyse a real dataset of DNA nucleotide sequences of A, G, C, T: the splice dataset [455] with $n = 3175$ nucleotide sequences, each belonging to one of three types: Exon–Intron junction (“EI”); Intron–Exon junction (“IE”); and Neither EI nor IE (“N”). Figure 35.5 summarises the results and shows our method’s impressive ability to extract biologically meaningful latent features. Here, we specify a relatively large number of deep latent classes with $B = 5$ in the Bayesian pyramid and impose a Dirichlet prior with hyperparameters $(0.01, 0.01, \dots, 0.01)$.

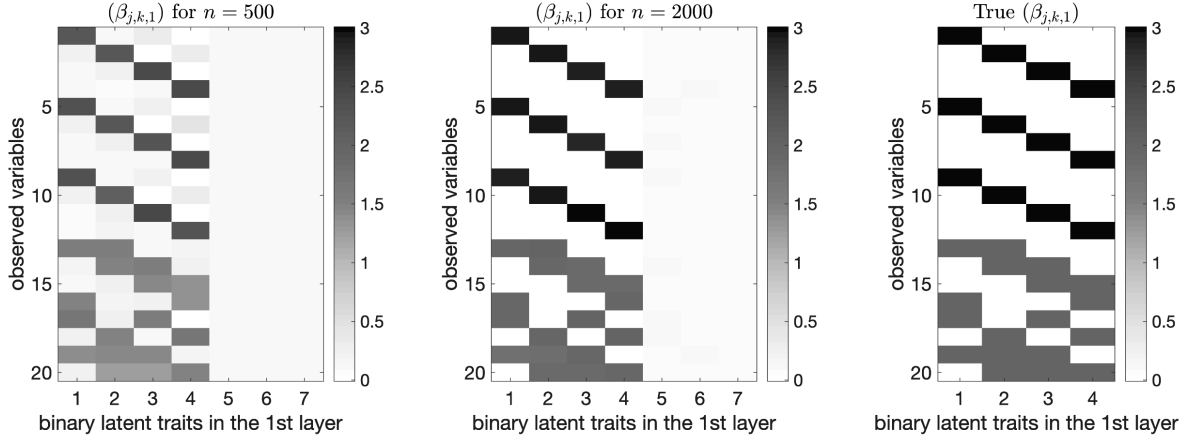


Figure 35.2: Posterior means of the Bayesian Pyramid parameters $\{\beta_{j1k}\}$ for response category $c = 1$ averaged across 50 independent replications, for $n = 500$ or 2000 . The value $K_{\text{upper}} = 7$ is used in the CUSP prior for posterior inference, while the true K_1 is 4 as shown in the rightmost plot.

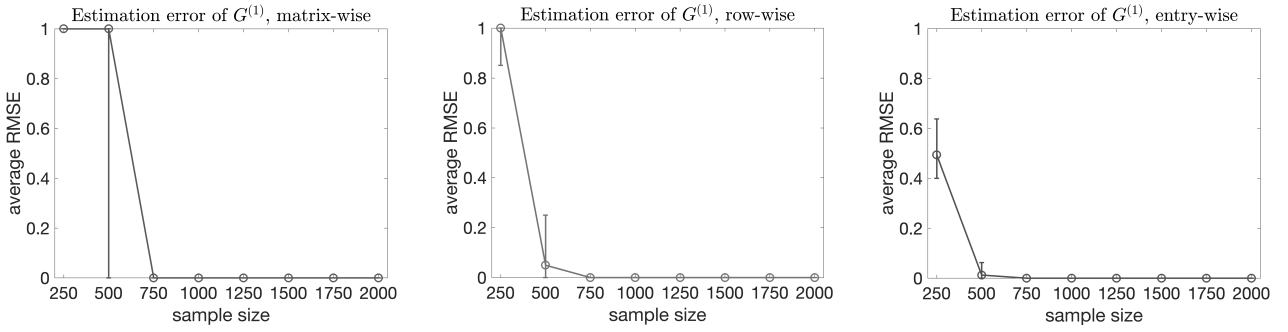


Figure 35.3: Mean estimation errors of the graphical matrix $\mathbf{G}^{(1)}$ at the matrix level (in (a)), row level (in (b)), and entry level (in (c)). The median, 25% quantile, and 75% quantile based on the 50 independent simulation replications are shown by the circle, lower bar, and upper bar, respectively.

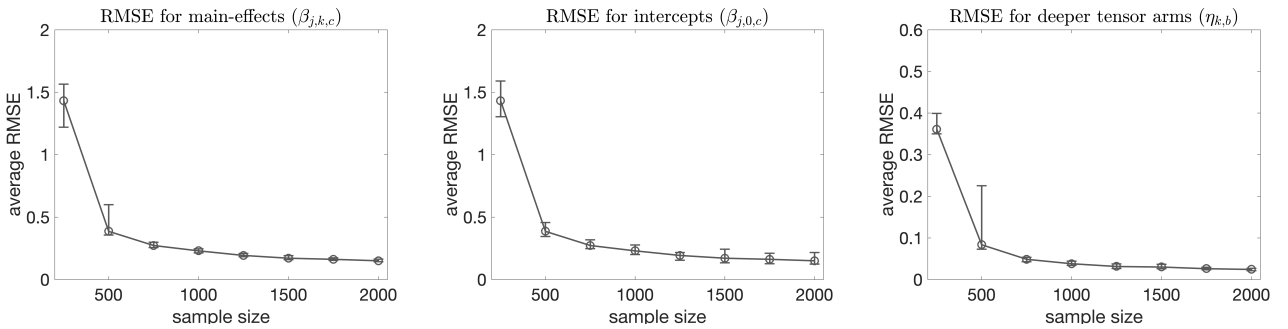


Figure 35.4: Average root-mean-square errors (RMSE) of the posterior means of the main effects β , intercepts β_0 , and deeper-layer conditional Bernoulli probability parameters η versus sample size $n = 250 \cdot i$ for $i \in \{1, 2, \dots, 8\}$. The median, 25% quantile, and 75% quantile based on the 50 independent simulation replications are shown by the circle, lower bar, and upper bar, respectively.

Interestingly, in this “overfitted” mixture setting, our algorithm is able to effectively empty out three out of five latent classes in the posterior, leading to $B_{\text{eff}} = 2$ meaningful latent classes that separate the EI type from the two other types. By visual inspection, one can already see the learned α -layer and z -layer latent features (left two panels of Figure 35.5) match the held-out sequence labels “EI”, “IE”, “N” provided by biologists (third panel) very well. A downstream classification based on a simple decision tree indeed yields very high accuracy, as shown in the confusion matrix in Figure 35.5. Such accuracy matches that given by fine-tuned convolutional neural networks

in Nguyen et al. [1244].

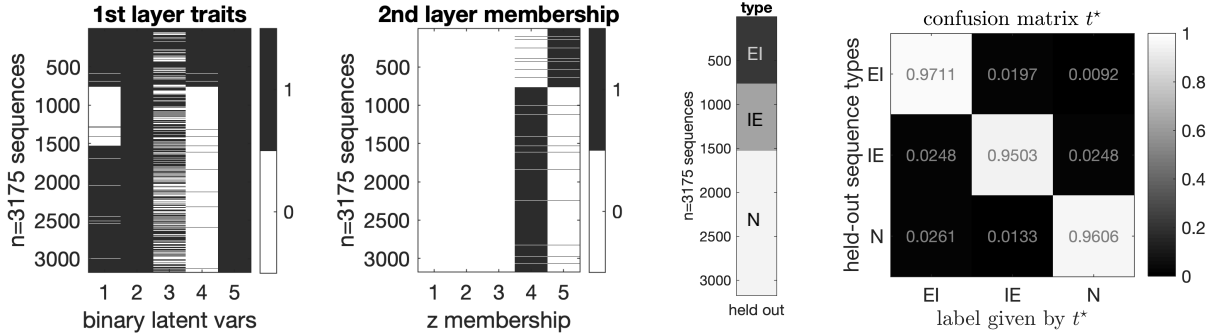


Figure 35.5: Application of two-latent-layer Bayesian Pyramid to DNA nucleotide sequence data [455]. Left to right: α -layer latent features; z -layer latent classes; held-out sequence labels; confusion matrix showing the downstream classification results.

35.3 Bayesian Latent Class Regression with Interpretable Binary Profiles

High-dimensional categorical data are prevalent in diverse scientific domains, including ecology, genetics, and the social sciences. For example, species co-occurrence across sampling sites is often represented by binary presence-absence vectors [1275, 1611], DNA sequences can be encoded as categorical strings of nucleotides [1883], and survey responses frequently capture high-dimensional patterns of individual choices and opinions [183]. These datasets are typically accompanied by covariates such as environmental conditions [1721], biological characteristics [28], or demographic information [1676], motivating the development of statistical models that jointly account for categorical outcomes and covariates. As one of the recent advances, Bayesian latent class regression with interpretable binary profiles [1877] leverages multilayer network structures to achieve theoretical guarantees on model identifiability and robust performance in high-dimensional settings. In an application to joint species distribution modelling in ecology, the model also exhibits performance comparable to that of state-of-the-art methods while improving the interpretability of the results.

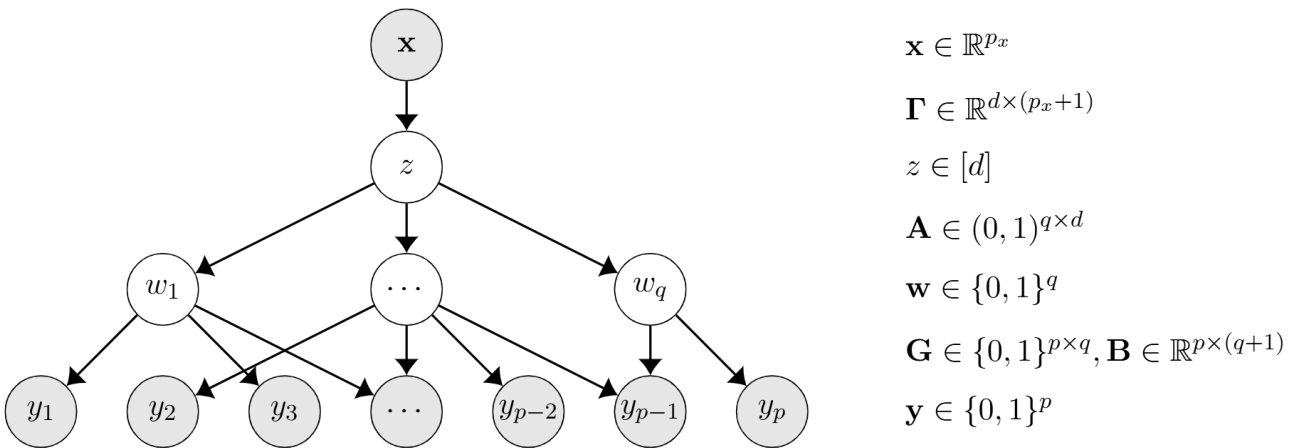


Figure 35.6: Bayesian latent class regression with interpretable binary profiles. Shading/unshading represents observed/latent variables. Covariates $\mathbf{x}$, deep latent class z , binary latent attributes $\mathbf{w}$, and observed variables $\mathbf{y}$ are specific to each sample. Parameters $\mathbf{A} = (\alpha_{j,z})$, $\mathbf{B} = (\beta_{i,j})$, $\mathbf{\Gamma} = (\gamma_{h,j})$, and $\mathbf{G} = (g_{i,j})$ are shared across the population and respectively encode the latent-attribute probabilities, observed-layer regression coefficients, covariate effects on the deep classes, and sparse dependency structure.

35.3.1 Model Structure and Identifiability Results

Compared to latent class regression [306, 382, 586, 651, 1121], Bayesian latent class regression with interpretable binary profiles (BLIP) introduces an additional layer of binary latent attributes $\mathbf{w}$ representing interpretable fine-grained traits, as illustrated by the network structure in Figure 35.6. The layer of binary latent attributes $\mathbf{w}$ lies between the layer of observed categorical outcomes $\mathbf{y}$ and the layer of deep latent classes z that captures broader population-level clusters influenced by covariates $\mathbf{x}$.

We let each observed categorical outcome dimension y_i be conditionally independent given the binary attributes $\mathbf{w} = (w_1, \dots, w_q)$, and further restrict each y_i to depend only on a subset of attributes w_j . This sparse dependency is introduced to improve model interpretability and characterised by a binary matrix $\mathbf{G}$, with its entry $g_{i,j} = 1$ indicating the dependency of y_i on w_j . Using a logit link function, the conditional distribution is modelled as

$$\log \frac{\mathbb{P}(y_i = 1 \mid \mathbf{w}, \mathbf{G}, \mathbf{B})}{\mathbb{P}(y_i = 0 \mid \mathbf{w}, \mathbf{G}, \mathbf{B})} = \beta_{i,0} + \sum_{j=1}^q g_{i,j} w_j \beta_{i,j}, \quad (35.9)$$

for logistic regression intercepts $\beta_{i,0}$ and coefficients $\beta_{i,j}$, and $\mathbf{B} := (\beta_{i,j})$. Building on this foundation, we structure the latent attributes $w_1, \dots, w_q$ by conditional independence on a deeper latent class $z \in \{1, \dots, d\}$ with Bernoulli distributions:

$$\mathbb{P}(\mathbf{w} \mid z, \mathbf{A}) = \prod_{j=1}^q \alpha_{j,z}^{w_j} (1 - \alpha_{j,z})^{1-w_j}, \quad (35.10)$$

for Bernoulli distribution parameters $\alpha_{j,z} \in (0, 1)$ and $\mathbf{A} := (\alpha_{j,z})$. To incorporate the influence of covariates $\mathbf{x}$, we further allow the deep latent class z to depend on $\mathbf{x}$ through a multinomial logistic regression with class probabilities

$$\mathbb{P}(z \mid \mathbf{x}, \mathbf{\Gamma}) = \frac{\exp(\gamma_{z,0} + \gamma_z^\top \mathbf{x})}{\sum_{h=1}^d \exp(\gamma_{h,0} + \gamma_h^\top \mathbf{x})}, \quad (35.11)$$

where $\gamma_{h,0}$ and γ_h serve as the regression intercept and coefficients for class h , respectively, and $\mathbf{\Gamma} := (\gamma_{h,0}, \gamma_h)$. To ensure identifiability, class d is designated as the baseline class with its regression intercept and coefficients fixed at zero.

The multilayer model structure improves model expressiveness compared to latent class regression while maintaining clear interpretability. The deep latent class z groups observations into d clusters, while the latent attributes w_j capture the presence or absence of fine-grained latent traits. According to the q latent attributes, the observations can be further partitioned into at most 2^q groups, providing a flexible and expressive cluster structure.

Although resembling the structure of a multilayer neural network, our model distinctively enjoys the property of identifiability. The sparsity $\mathbf{G}$ on the conditional dependency of y_i on w_j plays a pivotal role in establishing identifiability. As in latent class models, our model exhibits a trivial form of non-identifiability caused by label switching [693], arising from permutations over the d deep latent classes and permutations over the q latent attributes. A common practice to handle the label switching issue is through post-processing methods [1562].

Theorem 35.3. (1) *The model parameters are identifiable up to label switching, if the matrix $\mathbf{G}$ contains three distinct identity blocks $\mathbf{I}_q$ and the continuous parameters $\mathbf{A}, \mathbf{B}$ satisfy certain mild conditions.*

(2) *The model parameters are almost surely identifiable up to label switching, if $\mathbf{G}$ has distinct columns and contains three distinct blocks, two of which are $q \times q$ with all-ones diagonals and the third has no all-zeros columns.*

Theorem 35.3 establishes the identifiability of BLIP model under two notions: The stronger notion in (1) requires all parameters within the parameter space to be identifiable, while the weaker notion in (2) relaxes this requirement to almost every parameter in the parameter space with respect to the base measure. The weaker notion enjoys much weaker conditions on $\mathbf{G}$ while being sufficient for statistical inferences. To distinguish them, the stronger notion of identifiability is typically referred to as strict identifiability, whereas the weaker notion is referred to as generic identifiability. Mathematical definitions of strict/generic identifiability for BLIP model, together with a complete statement of Theorem 35.3 and its proof, can be found in Zhou et al. [1877].

35.3.2 Bayesian Inference

To perform Bayesian inference, we adopt independent weakly informative priors [596] by default, though structured priors can be applied in specific cases. Notably, using established identifiability results, the model is posterior

consistent under mild conditions on the prior distribution and parameter space, providing theoretical guarantees. A simple Gibbs sampler with Pólya–Gamma data augmentation [1337] can be used for posterior computation.

We have incorporated observation-specific covariates $\mathbf{x}$ that impact the distribution of the deep latent class z in (35.11). In many applications, there exists another type of covariates, which we refer to as *meta covariates* $\mathbf{t}_i$, which are associated with each dimension y_i of the observed data [1248]. For example, in ecology, each species has its biological traits; in genetics, each DNA nucleotide has a corresponding location; and in survey design, each question has its semantic features.

We design a hierarchical prior distribution on the sparse dependency matrix $\mathbf{G}$ that encourages y_i with similar meta covariates $\mathbf{t}_i$ to exhibit similar dependency structures on $\mathbf{w}$:

$$g_{i,j} \mid \mathbf{t}_i, \theta_{j,0}, \boldsymbol{\theta}_j \stackrel{\text{ind}}{\sim} \text{Bernoulli} \left(\frac{\exp(\theta_{j,0} + \boldsymbol{\theta}_j^\top \mathbf{t}_i)}{1 + \exp(\theta_{j,0} + \boldsymbol{\theta}_j^\top \mathbf{t}_i)} \right), \quad \theta_{j,0} \stackrel{\text{iid}}{\sim} N(0, 1), \quad \boldsymbol{\theta}_j \stackrel{\text{iid}}{\sim} N(0, \mathbf{I}). \quad (35.12)$$

Intuitively, we can interpret each $\boldsymbol{\theta}_j$ as a summary of the effects of the meta covariates relevant to w_j , such that a data entry y_i has a higher prior probability of depending on w_j when $\boldsymbol{\theta}_j^\top \mathbf{t}_i$ is larger.

35.3.3 Limiting Behaviour in High-dimensional Clustering

Although influential, traditional latent class models and their extensions such as mixed membership models [680, 1442] struggle in the high-dimensional regime, where these models are prone to over-fragmentation or collapsing to trivial clusters, a manifestation of the curse of dimensionality. Recent theoretical work shows that even well-specified Bayesian mixture models can asymptotically degenerate into singleton clusters as the data dimension grows [256, 1877]. For a motivating example, we consider the Bernoulli mixture model

$$\mathbb{P}_{\text{mix}}(\mathbf{y}^{(1:N)} \mid \mathbf{z}^{(1:N)}, \mathbf{a}) = \prod_{n=1}^N \prod_{i=1}^p \mathbb{P}_{\text{mix}}(y_i^{(n)} \mid z^{(n)}, a_{i,z^{(n)}}) = \prod_{n=1}^N \prod_{i=1}^p a_{i,z^{(n)}}^{y_i^{(n)}} (1 - a_{i,z^{(n)}})^{1-y_i^{(n)}} \quad (35.13)$$

with uniform prior distribution $a_{i,z} \sim \text{Uniform}(0, 1)$, and let the observations $\mathbf{y}^{(1:N)}$ be independent realisations from a single-cluster Bernoulli mixture model with parameters $a_{i,1} = \frac{1}{2}$.

Theorem 35.4 (Proposition 4.1 in Zhou et al. [1877]). *Let the number of observations N be fixed, and $z^{(1:N)} = \{\{1\}, \{2\}, \dots, \{N\}\}$ denote the partition into N separate clusters. As $p \rightarrow \infty$, we have $\mathbb{P}_{\text{mix}}(z^{(1:N)} = \{\{1\}, \{2\}, \dots, \{N\}\} \mid \mathbf{y}^{(1:N)}) \rightarrow 1$ almost surely.*

As shown by Theorem 35.4, our example of a well-specified Bernoulli mixture model with a single true cluster tends to assign each observation to a separate cluster when observations are high-dimensional. Multiple approaches can be designed to address this challenge faced by latent class regression and mixture models. BLIP model escapes the curse of dimensionality through the introduced layer of latent attributes of fixed dimension q . Intuitively, given high-dimensional observations $\mathbf{y}^{(1:N)}$, the posterior distribution of the deep latent classes $z^{(1:N)}$ is asymptotically equivalent to its conditional distribution given the true latent attributes $\mathbf{w}_*^{(1:N)}$ (referred to as the *Bayesian oracle clustering probability*).

Theorem 35.5 (Theorem 4.6 in Zhou et al. [1877]). *Under some mild conditions on well-specified data generating distribution and model prior distribution, we have $\lim_{p \rightarrow \infty} \mathbb{P}(z^{(1:N)} \mid \mathbf{x}^{(1:N)}, \mathbf{y}^{(1:N)}) = \mathbb{P}(z^{(1:N)} \mid \mathbf{x}^{(1:N)}, \mathbf{w}_*^{(1:N)})$ almost surely.*

The oracle probability $\mathbb{P}(z^{(1:N)} \mid \mathbf{x}^{(1:N)}, \mathbf{w}_*^{(1:N)})$ remains invariant and reliable regardless of the observed data dimensionality. Thus, for clustering high-dimensional binary data, Theorem 35.5 provides a theoretical guarantee of the reliability of posterior inference for deep latent classes $z^{(1:N)}$.

35.3.4 Application to Joint Species Distribution Modelling

As an application, we compare the traditional latent class regression (LCR) with our model (BLIP) in the setting of joint species distribution modelling (JSDM). JSDMs model the joint distribution of the occurrences of multiple species and their dependence on environmental covariates [1276–1279]. A widely used benchmark in JSDM research is the Finnish bird dataset [1275], which records the presence or absence of 50 bird species in various sampling locations in Finland over multiple years and includes environmental covariates for each sampling location, such as the mean temperature in April and May and habitat type, classified into broadleaf forests, coniferous forests,

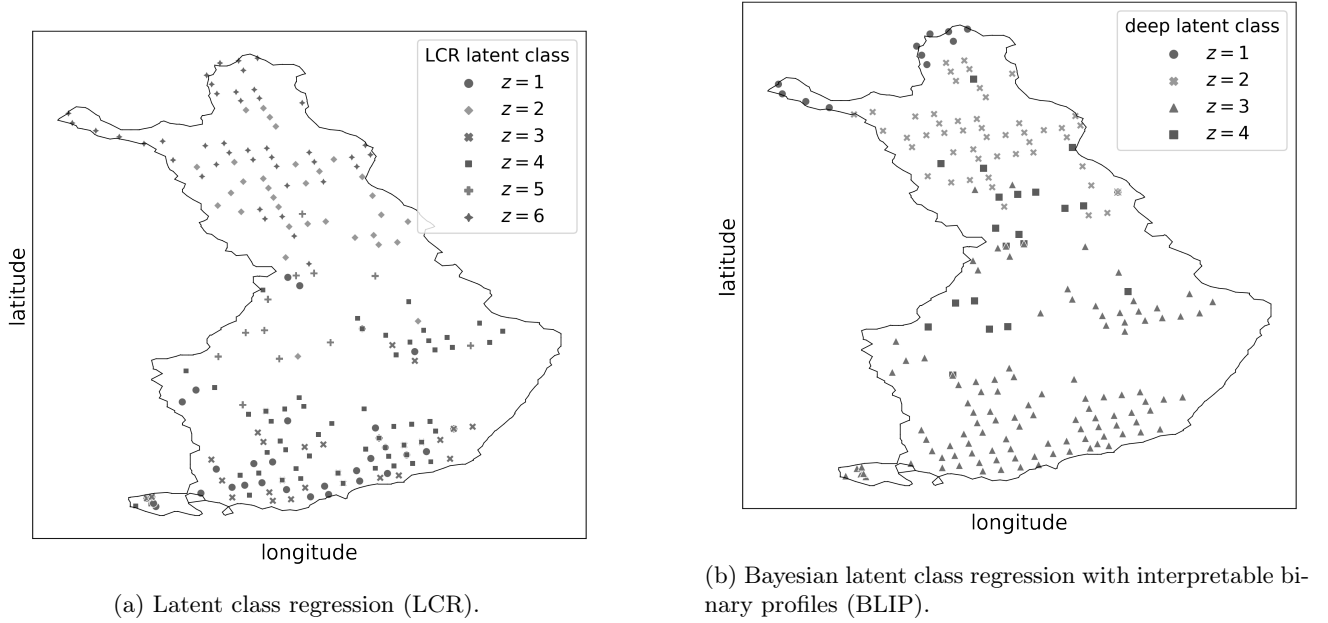


Figure 35.7: Spatial distributions of bird species sampling locations on the map of Finland, grouped by their inferred LCR latent class z or BLIP deep latent class z . In ecology, such clusters are referred to as regions of common profile.

open habitats, urban habitats, and wetlands. The dataset also has meta (species-specific) covariates including biological traits of each bird species, such as the log-transformed typical body mass and migration type (classified as long term, short term, and resident).

We used data recorded from 2011 to 2013 as in-sample observations and data recorded in year 2014 as out-of-sample observations, comprising 363 and 137 sampling locations, respectively. Using the Watanabe–Akaike information criterion [WAIC, 1724], we select the optimal model structure for BLIP with $q = 5$ latent attributes and $d = 4$ deep latent classes, and the optimal model structure for LCR with $d = 6$ latent classes. We visualise the (deep) latent class assignments of LCR and BLIP in Figure 35.7, plotted on the longitude–latitude plane of the corresponding sampling locations. Although both models do not incorporate spatial information, compared to LCR, the inferred deep latent classes in BLIP align much better with Finland’s geography, effectively capturing distinct ecological zones. Class $z = 1$ represents a borderline arctic tundra habitat, characterised by high mountains, the absence of trees, and a unique assemblage of bird species found nowhere else in the country. Class $z = 2$ corresponds to the northern boreal forest, which harbours Lapland forest species along with other old-growth forest species. Class $z = 3$ spans southern Finland, where a greater diversity of southern bird species is present. Finally, class $z = 4$ is concentrated along a small southwestern coastal region, which supports southern broadleaf forest species and potentially includes habitats for mire or peatland species.

35.4 Bayesian deep generative models for multiplex networks

Networks are a central object in many disciplines, from neuroscience and ecology to sociology and economics. An undirected network with p nodes is commonly represented by a symmetric binary adjacency matrix $\mathbf{X}$, where $X_{i,j} = 1$ if nodes i and j are connected. In many modern applications, however, we observe not just a single network but a collection of networks over the same set of nodes, so-called multiplex networks. Examples include brain connectivity networks for multiple individuals, trade networks between countries across different commodities, soccer passing networks across matches, or contact networks among students observed at different times.

Existing approaches often model a shared connection probability matrix with hierarchical factorisation, but rarely address simultaneous inference of node communities, low-dimensional network embeddings, and clustering of individual networks. Leveraging the idea of a multilayer discrete latent structure in a Bayesian pyramid, we designed a family of Bayesian deep generative models for multiplex networks [1876]. These models can simultaneously (1) recover hierarchical community structure of nodes at the population level, (2) capture heterogeneity across individual networks, and (3) provide multi-resolution, overlapping clustering representations of both nodes

and networks. Despite their multifunctionality, they are highly interpretable and enjoy identifiability properties established through recently introduced theoretical tools.

35.4.1 Model Structure and Identifiability Results



(a) Population Bayesian network $\mathcal{N}$ and connection matrices $\mathbf{A}_k$ between layers $(k-1)$ and k .

(b) Individual adjacency matrices $\mathbf{X}_k^{(1:N)}$ at each layer k .

Figure 35.8: An illustration of Bayesian deep generative model for multiplex networks, where the Bayesian network $\mathcal{N}$ in (a) has $(K+1) = 3$ layers with $p_k = 2(k+1)$ nodes in each layer k , and the sample in (b) has $N = 3$ individuals. Connection matrices: $\mathbf{A}_1 = (1, 0; 1, 0; 0, 1; 0, 1)$, and $\mathbf{A}_2 = (1, 0, 0, 0; 1, 0, 0, 0; 1, 1, 0, 0; 0, 1, 1, 1; 0, 0, 1, 1; 0, 0, 0, 1)$.

In this section, uppercase $\mathbf{X}$ denotes an adjacency matrix and is distinct from the lowercase covariate vector $\mathbf{x}$ used in the preceding section. In Figure 35.8, $\mathbf{X}_k^{(n)}$ denotes the adjacency matrix for individual n at layer k . We propose a multi-layer latent graphical model (Figure 35.8) that extends the Stochastic Block Model (SBM) framework to multiplex networks while also including a multiresolution structure. The key innovation is the introduction of latent adjacency matrices at successive resolutions:

- Population layer: nodes are grouped into communities via a binary membership matrix $\mathbf{A}$.
- Individual latent layers: for each network, a latent adjacency matrix $\tilde{\mathbf{X}}^{(n)}$ (introduced below) encodes which communities interact, allowing flexible variation across individuals.
- Observed layer: each network's adjacency matrix $\mathbf{X}^{(n)}$ is generated conditional on its latent adjacency structure.

This hierarchy can be repeated across multiple layers, forming a Bayesian pyramid in which higher layers capture the coarse community structure and lower layers capture finer details. Importantly, the directed edges in the Bayesian network represent statistical dependence, distinct from the observed network edges. The model naturally accommodates overlapping communities: each node can belong to multiple communities, and the probability of an edge between two nodes increases if they share communities or if their communities are adjacent in the latent structure. This leads to interpretable additive effects that distinguish shared versus cross-community interactions.

We focus on the two-layer version of the model and provide intuition by generalising step by step from a stochastic block model (SBM) [768]. A simple SBM for a single binary adjacency matrix $\mathbf{X} = \{X_{i,j}\}$ is as follows:

$$\mathbb{P}(X_{i,j} = 1 | \mathbf{A}, C, \gamma) = \frac{\exp\left(C + \gamma \sum_{s=1}^{\tilde{p}} A_{i,s} A_{j,s}\right)}{1 + \exp\left(C + \gamma \sum_{s=1}^{\tilde{p}} A_{i,s} A_{j,s}\right)}, \quad (35.14)$$

where $C, \gamma \in \mathbb{R}$ are parameters and the binary matrix $\mathbf{A} \in \mathbb{R}^{p \times \tilde{p}}$ denotes the community membership of the p nodes, so that node i belongs to the community s if and only if $A_{i,s} = 1$. For multiplex networks with N observed adjacency matrices $\mathbf{X}^{(1:N)} := \{\mathbf{X}^{(n)}\}_{n=1}^N$, the above model can be easily extended if we assume that the N networks share common community memberships $\mathbf{A}$ and that the networks are conditionally independent given these memberships. We leverage information from multiplex networks to improve flexibility over SBMs while characterising differences across the networks. With this in mind, we introduce *latent adjacency matrices* for each network, which characterise connections between communities: $\tilde{\mathbf{X}}^{(1:N)} := \{\tilde{\mathbf{X}}^{(n)}\}_{n=1}^N$. Each $\tilde{\mathbf{X}}^{(n)} = (\tilde{X}_{s,t}^{(n)})$ is a binary $\tilde{p} \times \tilde{p}$ matrix, where $\tilde{X}_{s,t}^{(n)} = 1$ or 0 indicates whether the latent community s and the latent community t are adjacent. In (35.14), the basic SBM assumes that the nodes are more likely to be adjacent if they belong

to the same community. We additionally assume that nodes in the n th individual network are more likely to be adjacent if they belong to adjacent communities in $\tilde{\mathbf{X}}^{(n)}$:

$$\mathbb{P}(X_{i,j}^{(n)} = 1 | \mathbf{A}, C, \mathbf{\Gamma}) = \frac{\exp\left(C + \mathbf{a}_i^\top (\tilde{\mathbf{X}}^{(n)} * \mathbf{\Gamma}) \mathbf{a}_j\right)}{1 + \exp\left(C + \mathbf{a}_i^\top (\tilde{\mathbf{X}}^{(n)} * \mathbf{\Gamma}) \mathbf{a}_j\right)}, \quad (35.15)$$

where $\mathbf{a}_i^\top \in \mathbb{R}^{1 \times \tilde{p}}$ denotes the i th row of $\mathbf{A}$, $*$ denotes the Hadamard product, and $\mathbf{\Gamma}$ is a symmetric $\tilde{p} \times \tilde{p}$ matrix with positive entries. As the latent adjacency matrices $\tilde{\mathbf{X}}^{(1:N)}$ form a sample of multiplex networks, the same approach of this subsection can be applied to model the distribution of the latent $\tilde{\mathbf{X}}^{(1:N)}$ and our model can be formulated hierarchically across multiple layers, forming a deep generative model for multiplex networks.

Interpretability hinges on identifiability. Similarly to Bayesian pyramids and Bayesian latent class regression with interpretable binary profiles, we establish two notions of identifiability for the family of Bayesian deep generative models for multiplex networks, notably strict identifiability ensuring that all parameters within the parameter space are identifiable up to label switching at each layer of the Bayesian network, and generic identifiability ensuring that almost every parameter within parameter space is identifiable up to label switching.

Theorem 35.6. (1) *The model parameters are identifiable up to label switching if the binary membership matrix $\mathbf{A}$ at each layer of the Bayesian network contains two distinct identity blocks.*

(2) *The model parameters are almost surely identifiable up to label switching if the binary matrix $\mathbf{A}$ at each layer of the Bayesian network contains two distinct blocks with all one diagonals and some additional mild conditions are satisfied.*

Formal statements of Theorem 35.6 and its proof can be found in Zhou et al. [1876]. While the proof uses Kruskal’s theorem on uniqueness of three-way CANDECOMP/PARAFAC (CP) tensor decomposition [937], as is standard for discrete latent variable models, we have also introduced a novel proof technique from complex analysis, which generalises existing techniques for proving identifiability and is of independent interest.

Theorem 35.7. *Let $f(z_1, \dots, z_n)$ be a holomorphic function with domain $\mathbb{C}^n$. If its set of zeros in $\mathbb{R}^n$ has positive Lebesgue measure, i.e.*

$$\lambda(\{x \in \mathbb{R}^n : f(x) = 0\}) > 0,$$

then $f \equiv 0$ on $\mathbb{C}^n$.

The new proof technique involves two steps, first establishing the set of non-identifiable parameters to be contained in the zero set of a holomorphic function (e.g. polynomial, rational, exponential, trigonometric functions, and their compositions), then showing the holomorphic function to be non-zero by finding a specific parameter that is identifiable.

35.4.2 Bayesian Inference

We impose conditionally independent and weakly informative prior distributions on the model parameters for simplicity. To encourage statistical parsimony and prevent each node from belonging to too many communities, we enforce sparsity in the binary membership matrices $\mathbf{A}$ through a hyperparameter S controlling the maximum number of communities allowed for each node. We recommend choosing S as 1 or 2 for interpretability and computational efficiency. Building on the identifiability results of the model, posterior consistency can also be established using classical Bayesian asymptotics theories [442, 1477], ensuring reliable inference as the sample size grows.

Posterior inference can be conducted through a Gibbs sampler with Pólya–Gamma data augmentation [1337]. To handle large-scale datasets, multiple techniques can be applied to accelerate posterior computation, including approximate MCMC methods such as a subsampling Gibbs sampler for speeding up the sampling step, and spectral initialisation methods leveraging the Mixed-SCORE algorithm for the degree-corrected mixed-membership (DCMM) model [845]. We treat the number of nodes in each layer of the Bayesian network as unknown hyperparameters and determine them through model selection based on the Watanabe-Akaike information criterion (WAIC) in Watanabe [1724].

35.4.3 Applications to Brain Connectivity Networks

We apply our model to brain structural connectome data from the Human Connectome Project (HCP) [487, 1664]. Following the Desikan atlas [416], the brain is divided into 68 anatomical regions of interest (ROI), and the observed data are the 68×68 adjacency matrices for each of the 1065 individuals.

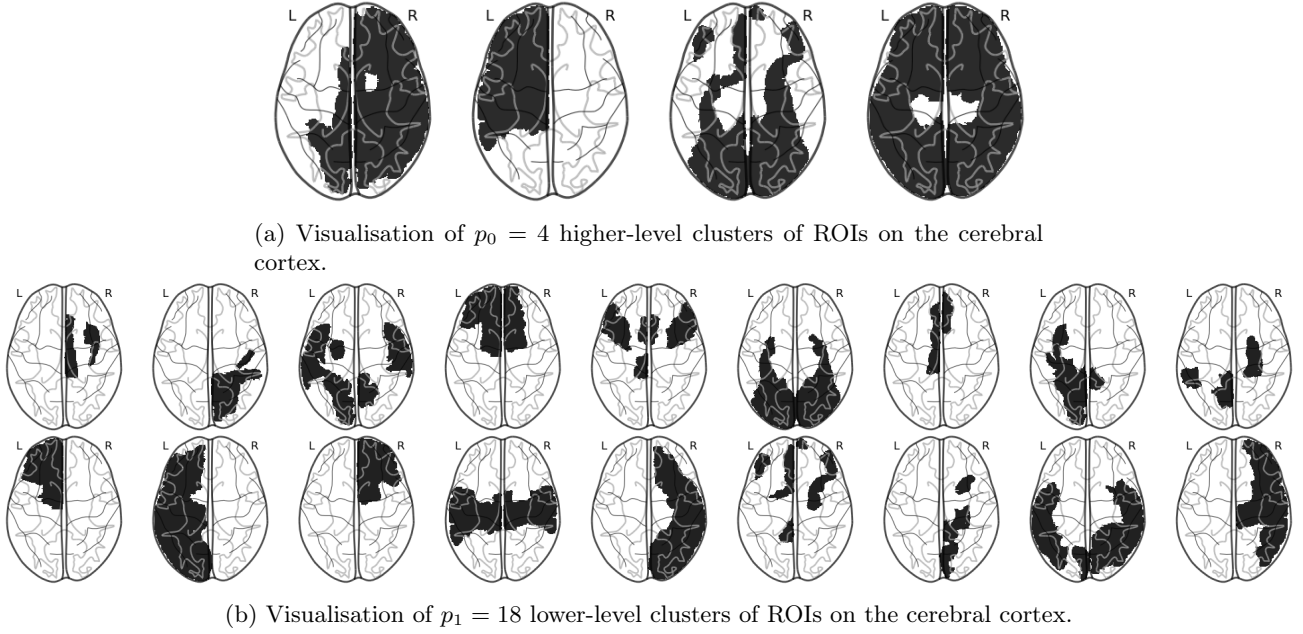


Figure 35.9: Posterior distributions of the higher-level and lower-level clusters given by connection matrices $\mathbf{A}$, under sparsity hyperparameter $S = 2$ and number of nodes in each Bayesian network layer $p_0 = 4, p_1 = 18, p_2 = 68$.

We consider applying the model with a three-layer Bayesian network, with the 68 nodes at the bottom layer being the brain regions of interest (ROIs). Under the sparsity parameter $S = 2$, the optimal model has 18 nodes in the middle layer and 4 nodes in the top layer. The 18 nodes in the middle layer can be viewed as communities for the 68 ROIs, and the 4 nodes in the top layer can be viewed as communities for the 18 nodes in the middle layer. The binary membership matrices $\mathbf{A}_1, \mathbf{A}_2$ indicate the memberships of these community assignments, forming a hierarchical overlapping clustering of the 68 ROIs.

We visualise the learned hierarchical overlapping clustering of ROIs on the brain cortex in Figure 35.9. The learned lower-level and higher-level clusters are highly interpretable, despite the model having no prior knowledge of where each ROI lies on the brain cortex. The lower-level clusters in Figure 35.9b form 18 overlapping regions on the cerebral cortex roughly corresponding to the frontal, parietal, temporal, and occipital lobes of the brain. Some of clusters are restricted to one hemisphere, while others include regions that are symmetric in the left and right hemispheres. In Figure 35.9a, among the four higher-level groups, two of them roughly correspond to the left and right hemispheres, while the remaining two are symmetric across the hemispheres roughly representing the temporal and occipital lobes, and all regions excluding the parietal lobe. We find that these clusters are roughly consistent with the known anatomical structure of the human brain even though we infer these spatially contiguous regions in a fully unsupervised manner without including information on spatial proximity or anatomy.

35.5 Going beyond Categorical Data to Arbitrary Data Types: Deep Discrete Encoders

Interestingly and somewhat surprisingly, the Bayesian Pyramids framework can be substantially generalised beyond categorical data to model *arbitrary types of data* while still enjoying identifiability. The Deep Discrete Encoders [DDEs, 2007] extend Bayesian Pyramids to model very rich types of data by allowing the observed data layer to follow any parametric distribution, while remaining fully identifiable under essentially the same structural conditions on the between-layer graphical matrices. A scalable two-stage computational pipeline is proposed in Lee and Gu [2007] to enable scaling DDEs to high-dimensional data and truly deep latent architectures with $D = 5$ latent layers. We apply DDEs to diverse real-world datasets with different data types (images, texts, response times) to perform hierarchical topic modelling, image representation learning, and multimodal modelling in educational testing. These applications demonstrate DDEs' flexibility, interpretability, representation power, and downstream prediction accuracy.

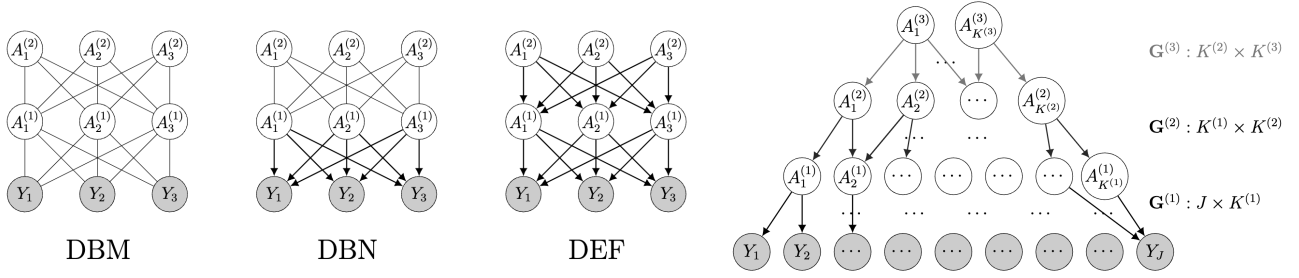


Figure 35.10: Graphical structures of DDEs and related existing DGMs: a deep Boltzmann machine (DBM), with binary data and undirected binary latent layers; a deep belief network (DBN), with binary data and directed binary latent layers apart from its top layer; a deep exponential family (DEF), with exponential-family conditional distributions; and a DDE (rightmost), with general responses Y_j and shrinking-ladder-shaped binary latent layers.

35.5.1 Model Setup of DDEs

DDEs have a flexible data layer at the bottom of the directed graphical model, but share the same key latent architecture as Bayesian Pyramids in that the discrete latent layers have progressively smaller sizes as they go deeper. Figure 35.10 illustrates DDEs together with some related DGMs. To distinguish the notation used for the two model families, Bayesian Pyramid latent layers are denoted by $\alpha^{(m)}$, whereas DDE latent layers are denoted by $\mathbf{A}^{(d)}$.

We next describe a DDE with D latent layers. The top layer latent variables are assumed to be independent Bernoullis with parameters $\mathbf{p} = (p_1, \dots, p_{K^{(D)}})$ to promote statistical parsimony:

$$\mathbb{P}(\mathbf{A}^{(D)} = \alpha^{(D)}) = \prod_{k=1}^{K^{(D)}} \mathbb{P}(A_k^{(D)} = \alpha_k^{(D)}) = \prod_{k=1}^{K^{(D)}} p_k^{\alpha_k^{(D)}} (1 - p_k)^{1 - \alpha_k^{(D)}}, \quad \forall \alpha^{(D)} \in \{0, 1\}^{K^{(D)}}. \quad (35.16)$$

For each $d > 1$, given $\mathbf{A}^{(d)}$, we define the $(d-1)$ th layer by assuming the conditional independence of $A_1^{(d-1)}, \dots, A_{K^{(d-1)}}^{(d-1)}$ given $\mathbf{A}^{(d)}$. Each conditional distribution is

$$A_k^{(d-1)} \mid \mathbf{A}^{(d)} \sim \text{Bernoulli}\left(g_{\text{logistic}}\left(\beta_{k,0}^{(d)} + \sum_{l=1}^{K^{(d)}} g_{k,l}^{(d)} \beta_{k,l}^{(d)} A_l^{(d)}\right)\right), \quad (35.17)$$

where $g_{\text{logistic}}(x) = 1/(1 + e^{-x})$; one may use alternative link functions $g : \mathbb{R} \rightarrow [0, 1]$, such as the probit link. Here, $g_{k,l}^{(d)} = 1$ or 0 is the binary indicator of whether $A_l^{(d)}$ is a parent of $A_k^{(d-1)}$. Similarly as in Bayesian Pyramids, we collect these indicators into a graphical matrix $\mathbf{G}^{(d)} \in \{0, 1\}^{K^{(d-1)} \times K^{(d)}}$ for each d , which summarises the bipartite graph structure between the $(d-1)$ th and d th latent layer. Collect the $\beta_{k,l}$ -parameters in a $K^{(d-1)} \times (K^{(d)} + 1)$ matrix $\mathbf{B}^{(d)}$, whose first column is the intercepts $(\beta_{k,0}^{(d)})_{k \in [K^{(d-1)}]}$ and remaining parts are $(\beta_{k,l}^{(d)})_{k \in [K^{(d-1)}], l \in [K^{(d)}]}$.

Finally, we model the bottom layer for the observed data. The observed variables $\mathbf{Y} = (Y_1, \dots, Y_J)$ are modelled by assuming the conditional independence of $Y_1, \dots, Y_J$ given $\mathbf{A}^{(1)}$. As the data are not necessarily categorical, we consider a general parametric family of the form

$$Y_j \mid \mathbf{A}^{(1)} \sim \text{ParFam}_j\left(g_j\left(\beta_{j,0}^{(1)} + \sum_{k=1}^{K^{(1)}} g_{j,k}^{(1)} \beta_{j,k}^{(1)} A_k^{(1)}, \gamma_j\right)\right). \quad (35.18)$$

Here, ParFam_j denotes an arbitrary parametric family, and H_j denotes its parameter space. Let p_j be the probability mass/density function of ParFam_j . The known link function $g_j : \mathbb{R} \times [0, \infty) \rightarrow H_j$ maps the linear predictor and, when present, the dispersion parameter $\gamma_j > 0$ to the parameters of the given family; the dispersion argument is omitted for families that do not use it. Similarly, denote $\mathbf{G}^{(1)} = (g_{j,k}^{(1)})_{j \in [J], k \in [K^{(1)}]}$ as the graphical matrix between the observed layer and the shallowest latent layer.

Next, we give some examples of the various response types $\mathcal{Y}_j$ allowed in the DDE framework, along with the corresponding link functions g and parametrisations for (35.18). Consider three types of responses for example: (i) binary, (ii) count, and (iii) continuous. We can model each of these responses using (i) Bernoulli with $g = g_{\text{logistic}}$, (ii) Poisson with an exponential link $g(x) = e^x$, and (iii) Normal with an identity link $g(x, y) = (x, y)$, respectively. Modelling other data types is also straightforward. We remark that the DDE framework can easily handle the case where the observed variables at the bottom layer have different data types and follow different distributions.

35.5.2 Identifiability and Estimation Consistency of DDEs

Lee and Gu [1007] show that DDEs are identifiable under essentially the same structural conditions on the graphical matrices $\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(d)}$ as Bayesian Pyramids, even under challenging and flexible settings with arbitrary parametric distribution families employed in the bottom layer. The following theorem establishes the identifiability of DDEs.

Theorem 35.8 (Theorem 1 in Lee and Gu [1007]). *Consider a D -latent-layer DDE where the number of latent variables is given. Suppose that for all $d \leq D-1$, the true graphical structures $\mathbf{G}^{(d)*}$ and parameters $\mathbf{B}^{(d)*}$ satisfy the following conditions:*

- A. *Each latent variable $A_k^{(d)}$ has at least two pure children $A_{j_{k,1}}^{(d-1)}, A_{j_{k,2}}^{(d-1)}$ according to $\mathbf{G}^{(d)*}$.*
- B. *For any $\alpha^{(d)} \neq \alpha'^{(d)} \in \{0, 1\}^{K^{(d)}}$, there exists $j \in [K^{(d-1)}] \setminus \cup_{k=1}^{K^{(d)}} \{j_{k,1}, j_{k,2}\}$ such that $\sum_{k=1}^{K^{(d)}} \beta_{j,k}^{(d)*} (\alpha_k^{(d)} - \alpha'_k{}^{(d)}) \neq 0$.*

Then, the model is strictly identifiable.

The key insight behind proving the identifiability of DDEs for rich data types is that, with discrete latent layers, it suffices to establish identifiability for a one-latent-layer model and proceed in a layer-wise manner inductively. To elaborate, consider a “collapsed” DDE where the latent layers indexed by $d = 2, \dots, D$ are marginalised out to give a probability mass function (pmf) for the first latent layer: $\{\mathbb{P}(\mathbf{A}^{(1)} = \alpha^{(1)}) : \alpha^{(1)} \in \{0, 1\}^{K^{(1)}}\}$. If the collapsed one-layer model is proven to be identifiable, that means the conditional distributions $\mathbb{P}(Y_j | \mathbf{A}^{(1)})$ and the marginal distribution $\mathbb{P}(\mathbf{A}^{(1)})$ can be uniquely identified from the data distribution $\mathbb{P}(\mathbf{Y})$. Then, we can theoretically treat the shallowest latent layer $\mathbf{A}^{(1)}$ as if it were observed, because its pmf is now identified and known. Then, viewing this pmf as the marginal distribution of $K^{(1)}$ -dimensional observed variables of a $(D-1)$ -latent-layer Bernoulli-DDE, we can inductively identify all parameters via a layerwise argument. Notably, with a flexible data layer in DDEs, the first step of proving the collapsed model to be identifiable involves the following key technical step: carefully categorising the sample spaces of the general responses Y_j and defining surrogate categorical responses $\tilde{Y}_j$, whose joint distribution can be represented by a probability tensor. Then we invoke the classical multilinear algebra result on the uniqueness of tensor decompositions [937] to establish identifiability.

We next state the generic identifiability result for DDEs. In graph theory, a bipartite graph is said to have a “perfect matching” if it contains a set of edges without common vertices that covers every vertex of the graph [703].

Theorem 35.9 (Theorem 2 in Lee and Gu [1007]). *Consider the D -latent-layer DDE where the number of latent variables $\mathcal{K}$ is given, and all parametric families and link functions g_j are analytic. Let $\mathcal{G}^*$ denote the true graphical matrices and suppose the true parameter lives in the graph-constrained parameter space $\Omega_{\mathcal{K}}(\Theta; \mathcal{G}^*)$. Suppose that for all $d \leq D-1$, the true graphical structure $\mathbf{G}^{(d)}$ satisfies the following condition C:*

- C. *There exists a partition of the $(d-1)$ th layer variable indices $[K^{(d-1)}] = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3$ that satisfies the following properties: (i) for $a = 1, 2$, there is a perfect matching in the bipartite graph between $\mathbf{A}_{\mathcal{I}_a}^{(d-1)}$ and $\mathbf{A}^{(d)}$, (ii) each latent variable $A_k^{(d)}$ has at least one child in $\mathbf{A}_{\mathcal{I}_3}^{(d-1)}$.*

Then, the model components $(\Theta, \mathcal{G})$ are generically identifiable.

Estimation Consistency. We propose a penalised maximum likelihood estimation method for DDEs and show that the estimator is consistent for both the continuous parameters and the discrete graph structures. Suppose that the numbers of latent variables in all layers are known. We maximise the following objective function to estimate parameters $\Theta = (\mathbf{p}, \mathcal{B}, \gamma)$:

$$\hat{\Theta} \in \arg \max_{\Theta} \left[\ell(\Theta | \mathbf{Y}) - \sum_{d=1}^D p_{\lambda_N, \tau_N}(\mathbf{B}^{(d)}) \right], \quad (35.19)$$

where $\ell(\Theta | \mathbf{Y}) = \sum_{i=1}^N \log \mathbb{P}(\mathbf{Y}_i | \Theta)$ denotes the marginal log-likelihood function given a sample $\mathbf{Y}_1, \dots, \mathbf{Y}_N$ of size N , defined based on the marginal distribution of $\mathbf{Y}$. The term $p_{\lambda_N, \tau_N} : \mathbb{R} \rightarrow [0, \infty)$ denotes a sparsity-inducing penalty, such as the truncated lasso penalty (TLP) [1509] and smoothly clipped absolute deviation (SCAD) penalty [504]. We slightly abuse notation and let $\mathbf{Y}$ denote the $N \times J$ data matrix including $\mathbf{Y}_1, \dots, \mathbf{Y}_N$ as rows. Using

the estimated coefficients in $\hat{\mathcal{B}} = (\hat{\beta}_{l,k}^{(d)})$, the layer-wise graphical matrices in $\mathcal{G}$ can be estimated by reading off the sparsity pattern of $\hat{\mathcal{B}}$:

$$\hat{g}_{l,k}^{(d)} := 1_{\hat{\beta}_{l,k}^{(d)} \neq 0} \quad \text{for all } d \in [D], k \in [K^{(d)}], l \in [K^{(d-1)}]. \quad (35.20)$$

Assuming a compact parameter space with bounded coefficients $\mathbf{B}^{(d)}$, the next theorem shows that the estimator defined in (35.19) and (35.20) results in consistent estimation.

Theorem 35.10 (Theorem 3 in Lee and Gu [1007]). *Consider a D -latent-layer DDE with true parameters $\Theta^*, \mathcal{G}^*$ and known $\mathcal{K}$. Assume that the model at Θ^* is identifiable, has a non-singular Fisher information, and all entries of $\{\mathbf{B}^{(d)}\}_{d=1}^D$ are bounded. Then, the estimator $\hat{\Theta}$ in (35.19) is $\sqrt{N}$ -consistent in the sense that there exists some $\tilde{\Theta} \sim_{\mathcal{K}} \hat{\Theta}$ such that $\|\tilde{\Theta} - \Theta^*\| = O_p(1/\sqrt{N})$. Here, $\sim_{\mathcal{K}}$ denotes equivalence up to within-layer permutations of the latent variables, and $\|\cdot\|$ denotes the vectorised L^2 norm. Additionally, the graphical matrices are consistently estimated: for $\tilde{\mathcal{G}}$ resulting from $\tilde{\Theta}$ according to (35.20), we have $\mathbb{P}(\tilde{\mathcal{G}} \neq \mathcal{G}^*) \rightarrow 0$.*

35.5.3 Scalable Computation Pipeline

We present a two-stage scalable computational pipeline to compute the penalised maximum likelihood estimator in (35.19). This method builds upon the penalised expectation–maximisation (EM) algorithm by including a spectral initialisation and stochastic approximation. For notational simplicity, we describe the method for $D = 2$, which straightforwardly extends to an arbitrary number of latent layers.

Stage One: Layerwise Double-SVD Initialisation. The multiple layers of nonlinearity in DDEs lead to a highly nonconvex optimisation landscape with potentially exponentially many local optima. Hence, it is critical to initialise the optimisation algorithm wisely. Spectral methods have been widely used for initialising linear low-rank matrix factorisation, but existing approaches are not applicable for DDEs with a deep nonlinear structure. To address this, we propose a novel layerwise nonlinear spectral initialisation strategy, outlined in Algorithm 29.

Algorithm 29: Outline of the Layerwise Double-SVD Initialisation

Data: Data matrix $\mathbf{Y}_{N \times J}$, latent dimensions $\mathcal{K}$.

Result: Initial layerwise estimates $\{\mathbf{B}^{(d)}, \mathbf{A}^{(d)}\}_{d=1}^D$.

1. De-noise the data matrix $\mathbf{Y}$ using a first SVD, and linearise this matrix by applying the inverse-link function $(\mu \circ g)^{-1}$ elementwise. Let $\hat{\mathbf{Z}}$ denote the inverted matrix.
 2. Let $\hat{\mathbf{Z}}_0$ be the column-centred version of $\hat{\mathbf{Z}}$, and compute its rank- $K^{(1)}$ approximation by a second SVD: $\hat{\mathbf{Z}}_0 \approx \mathbf{U}_{N \times K^{(1)}} \boldsymbol{\Sigma}_{K^{(1)} \times K^{(1)}} \mathbf{V}_{J \times K^{(1)}}^\top$.
 3. Rotate $\mathbf{V}$ according to the Varimax criteria, and denote it as $\hat{\mathbf{B}}^{(1)}$.
 4. Use the relationship $\hat{\mathbf{Z}} \approx [\mathbf{1}_N, \mathbf{A}^{(1)}] \mathbf{B}^{(1)\top}$ to estimate $\mathbf{A}^{(1)}$.
 5. Now, suppose that the estimated $\hat{\mathbf{A}}^{(1)}$ is the “observed data” of a one-latent-layer DDE. Repeat steps 1-4 to estimate $\mathbf{B}^{(2)}$ and $\mathbf{A}^{(2)}$.
-

The main idea of Algorithm 29 is to view the responses $\mathbf{Y}$ as a perturbation of a population expectation $\mathbb{E}(\mathbf{Y} \mid \mathbf{A}^{(1)}, \mathbf{B}^{(1)}) = (\mu \circ g) ([\mathbf{1}_N, \mathbf{A}^{(1)}] \mathbf{B}^{(1)\top})$, which is an elementwise (nonlinear) transformation of a low-rank matrix. Here, $\mu : H \rightarrow \mathcal{Y}$ is the known mean function of the observed-layer parametric family in (35.18) and $g : \mathbb{R} \rightarrow H$ is the link function. The function $(\mu \circ g)$ is g_{logistic} for Bernoulli responses, the exponential function for Poisson, and the identity function for Normal. When $(\mu \circ g)$ is nonlinear, we use the double SVD procedure [1846]. This procedure applies a first SVD to de-noise the data, then linearises the data by inverting the link function $(\mu \circ g)$, and finally performs a second SVD to find the low-rank matrix $\hat{\mathbf{Z}} \approx [\mathbf{1}_N, \mathbf{A}^{(1)}] \mathbf{B}^{(1)\top}$. Next, noting that the true coefficient matrix $\mathbf{B}^{(1)}$ is sparse, we estimate it by seeking a sparse rotation of the right singular subspace of $\hat{\mathbf{Z}}$, using the popular Varimax criterion [860, 1416]. This procedure also provides an estimate of the latent $\mathbf{A}^{(1)}$, which can be treated as the “observed data” to initialise the deeper layer’s $\mathbf{B}^{(2)}$ and $\mathbf{A}^{(2)}$ in a similar fashion as described above. This layer-by-layer algorithm readily generalises to deeper models and is reminiscent of the greedy learning strategy for DBNs [752].

Stage Two: Penalised SAEM Algorithm. Here t indexes the iterations, so $\Theta^{[t]}$ denotes the parameter estimate at iteration t . For DDEs with a large number of latent variables, the E-step in standard EM algorithms computes all conditional probabilities $\mathbb{P}(\mathbf{A}_i^{(1)} = \alpha^{(1)}, \mathbf{A}_i^{(2)} = \alpha^{(2)} \mid \mathbf{Y}; \Theta^{[t]})$ for all $\alpha^{(1)} \in \{0, 1\}^{K^{(1)}}$ and $\alpha^{(2)} \in \{0, 1\}^{K^{(2)}}$. This requires computing and storing $O(N \times 2^{\sum_{d=1}^D K^{(d)}})$ terms. The exponential dependency in $K^{(d)}$ is concerning even for moderately large latent dimensions, say $K^{(d)} = 10$, and quickly becomes prohibitive for larger $K^{(d)}$. Therefore, we propose a penalised stochastic approximation expectation–maximisation (SAEM) algorithm [see 406] by modifying both the E-step and M-step to more scalable versions using approximate sampling. This method has computing-time and memory costs that depend linearly on $\sum_{d=1}^D K^{(d)}$.

We elaborate on the details on deriving the SAEM. *First*, we replace the standard E-step with a simulation step, which simulates only a small number (denoted as C) of posterior samples of the latent variables. As exact sampling from the joint distribution $\mathbb{P}(\mathbf{A}_i^{(1)}, \mathbf{A}_i^{(2)} \mid \mathbf{Y}; \Theta^{[t]})$ is expensive, we sample each latent variable from their complete conditionals: sample each $A_{i,k}^{(1),[t+1]}$ from the Bernoulli distribution $\mathbb{P}(A_{i,k}^{(1)} \mid (-), \Theta^{[t]})$, where $(-)$ denotes the estimates of all other latent variables from the t th iteration. This simulation-based E-step drastically reduces the exponential complexity to a linear complexity by computing and storing only $O(N \times \sum_{d=1}^D K^{(d)})$ terms. We empirically find that taking $C = 1$ sample in each iteration is computationally efficient without sacrificing much accuracy. *Second*, we modify the standard M-step objective function (expected complete data log-likelihood) by (i) replacing the exact conditional probability values with sample-based quantities, (ii) stochastically averaging the objective functions, and (iii) including the sparsity-inducing penalties in (35.19). Algorithm 30 summarises the SAEM algorithm with $C = 1$. Here, the detailed M-step updates can be written in terms of low-dimensional maximisations for each row of the coefficient matrices.

Although the above two-stage pipeline is not a Bayesian inference procedure, one could easily develop a Bayesian version of DDEs and adapt the full conditional distributions derived in the SAEM Algorithm 30 to develop a Gibbs sampler and apply it if there is enough computing budget in high-dimensional scenarios. In this case, one can still first run the layerwise nonlinear spectral initialisation in Algorithm 29 to initialise the MCMC chain and speed up convergence.

Algorithm 30: Penalised SAEM algorithm for the two-latent-layer DDE

Data: $\mathbf{Y}, \mathcal{K}$, tuning parameters λ_N, τ_N .

Initialise $\mathbf{A}^{(1),[0]}, \mathbf{A}^{(2),[0]}$ and $\Theta^{[0]}$ using Algorithm 29.

while $\|\Theta^{[t]} - \Theta^{[t-1]}\|$ is larger than a threshold **do**

In the t th iteration, // **Simulation-step** Sample each $A_{i,k}^{(1),[t+1]}, A_{i,l}^{(2),[t+1]}$ from the complete conditionals using the previous parameter estimates $\Theta^{[t]}, \mathbf{A}^{[t]}$ // **Stochastic approximation**

M-step update the parameters $\Theta^{[t+1]}$ by maximising the stochastically averaged objectives

Estimate $\{\mathbf{G}^{(d)}\}_{d \in [D]}$ based on the sparsity structure of $\hat{\mathbf{B}}$ according to (35.20).

Output: Estimated continuous parameters $\hat{\Theta}$, estimated graphical matrices $\mathcal{G}$.

35.5.4 Simulation Results

For relatively shallow DDEs with $D = 2$ latent layers, the simulation results deliver a similar message as those for Bayesian Pyramids – they empirically confirm the identifiability theory and demonstrate the excellent performance of the proposed algorithms. We thus omit the simulation results in those standard regimes, but present results under deeper models and ablation studies.

Simulation studies under deep models. We assess the scalability and estimation accuracy of the two-stage computational pipeline for *deeper models* (with $D = 3, 4, 5$ latent layers) with potentially *high-dimensional responses* (with the observed data dimension set to $J = 486$ in the $D = 5$ case). For all settings, we consider Normal observations with $K^{(D)} = 2$ latent variables for the top (deepest) layer and $K^{(d-1)} = 3K^{(d)}$ variables for the d th layer for each $d = D, \dots, 2$.

Table 35.1 summarises the estimation accuracy of the graphs in each layer and the average runtime for DDEs with $D = 5$ latent layers. The computation time illustrates that our method is scalable to deeper models. Specifically, the $D = 3, 4$ cases only took a few minutes (tables omitted here), and the most challenging case with $D = 5$ and $N = 16,000$ also took less than an hour on a personal laptop. The estimation accuracy is higher for shallower layers than for deeper layers. This results from the accumulation of uncertainty for inferring deeper

layers, which is inevitable in hierarchical models as each layer consists of stochastic latent variables. Notably, even for deep models with $D = 5$, the shallow structures $\mathbf{G}^{(1)}, \mathbf{G}^{(2)}$ are still recovered with high accuracy. This indicates that one can include additional latent layers to increase the models' representational power, without sacrificing the accuracy of learning graph structures closer to the data layer.

Layer \ N	500	1000	2000	4000	8000	16000
$\mathbf{G}^{(1)}$	1.00	1.00	1.00	1.00	1.00	1.00
$\mathbf{G}^{(2)}$	0.81	0.90	0.95	0.97	0.98	0.98
$\mathbf{G}^{(3)}$	0.77	0.83	0.86	0.88	0.90	0.92
$\mathbf{G}^{(4)}$	0.61	0.62	0.65	0.67	0.66	0.74
$\mathbf{G}^{(5)}$	0.60	0.59	0.62	0.62	0.65	0.64
runtime (s)	220	259	326	563	893	2925

Table 35.1: Average entrywise-accuracy of estimating the graphical matrices and runtime in seconds for DDEs with $D = 5$ latent layers.

Ablation studies. As DDEs closely resemble DBNs [752], we compare DDEs with DBNs. For a fair comparison, we considered two settings: (a) data generated from a Bernoulli-DDE, and (b) data generated from a DBN with identical coefficients. The results in Figure 35.11 demonstrate that our proposed algorithm has better estimation accuracy for both the graphical structure and the continuous parameters under both well-specified and misspecified settings, even when data are generated from a DBN with undirected edges in the top layer. This is because the DBN algorithm does not learn sparse coefficients and fundamentally suffers from local optima due to random initialisation. The ablation studies indicate that the current implementations of DDEs are slower than DBNs. We believe that our DDE algorithms can be further scaled up by replacing the M-step in the SAEM algorithm with first-order optimisation methods (e.g. gradient ascent or stochastic gradient ascent) in future studies.

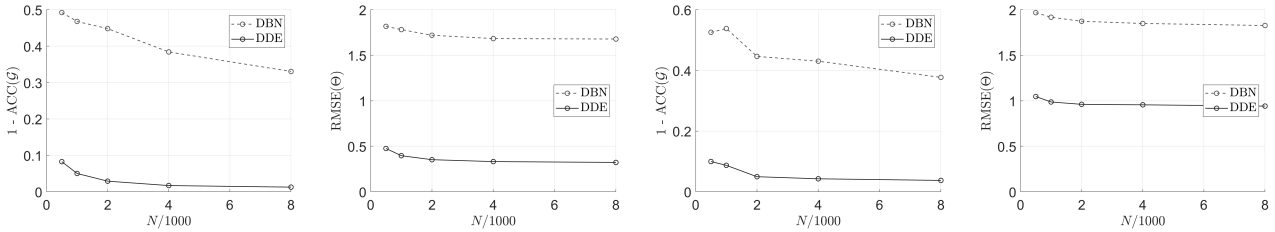


Figure 35.11: Comparison of Bernoulli-DDE versus DBN. True data are generated from a **DDE** in (a)–(b), and from a **DBN** in (c)–(d). Dashed and solid lines show results obtained by the DBN algorithm and our DDE algorithm, respectively. (a) and (c): estimation error for the graphs $\mathcal{G}$. (b) and (d): estimation error for continuous parameters Θ . Smaller values are better.

35.5.5 Diverse Real-world Applications: Images, Texts, Multimodal Data

Binary Data: Bernoulli-DDE for MNIST Handwritten Digits. The MNIST dataset for handwritten digits is very popular for classification as well as unsupervised learning [409]. We fit the two-latent-layer DDE with binary responses (Bernoulli-DDE), where the observed layer distributions are $\mathbf{Y}_j \mid \boldsymbol{\alpha}^{(1)} \sim \text{Ber}(g_{\text{logistic}}(\beta_{j,0}^{(1)} + \sum_{k \in [K^{(1)}]} \beta_{j,k}^{(1)} \alpha_k^{(1)}))$. This resembles existing generative models for images such as DBNs and DBMs, but we instead consider an identifiable, much lower-dimensional, shrinking-ladder-shaped latent structure. For easier presentation, we consider the subset of images whose true digit labels are 0, 1, 2, and 3. After preprocessing, our training set consists of $N = 20,679$ images, each with $J = 264$ binary pixels, representing challenging and relatively high-dimensional binary data. Compared to many existing works analysing MNIST, we consider a more challenging fully unsupervised setting by holding out all other information about the images, such as the true labels, number of classes, and the spatial location among the pixels.

We fit the two-latent-layer DDE with the latent dimensions set to $K^{(1)} = 5$ and $K^{(2)} = 2$. Figure 35.12 displays the first-layer coefficients $\mathbf{B}^{(1)}$ by rearranging each column into the original 28×28 grid. Note that the value of $\mathbf{B}^{(1)}$ is on the logit scale, so the negative coefficients make the corresponding pixel more likely to be zero. From

the reshaped columns $\beta_{0,0}^{(1)}, \beta_{0,1}^{(1)}, \dots, \beta_{0,5}^{(1)}$ of $\mathbf{B}^{(1)}$, we can interpret the meaning of each latent variable: $A_1^{(1)} = 1$ indicates a zero-like shape, $A_5^{(1)} = 1$ indicates symmetric curves on the upper-left and bottom-right corners, and the other latent variables represent different rotations. Additionally, using the estimated deeper graphical matrix $\mathbf{G}^{(2)}$, we can interpret the top layer latent variables as broader information about the images: e.g., $A_1^{(2)}$ indicates large pixel density and $A_2^{(2)}$ indicates symmetry with respect to the x-axis.

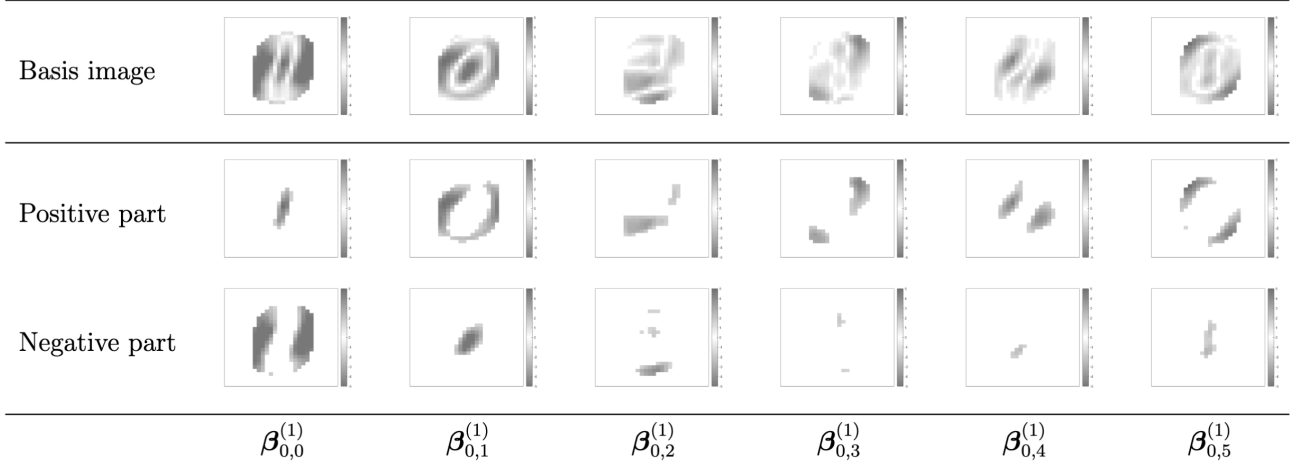


Figure 35.12: DDE-estimated basis images (reshaped from the estimated $\beta_{0,k}^{(1)}$) from the MNIST data, which define sparse subregions in the 28×28 image. The second and third rows show the most significant positive and negative parts of the basis images after thresholding at ± 1.5 .

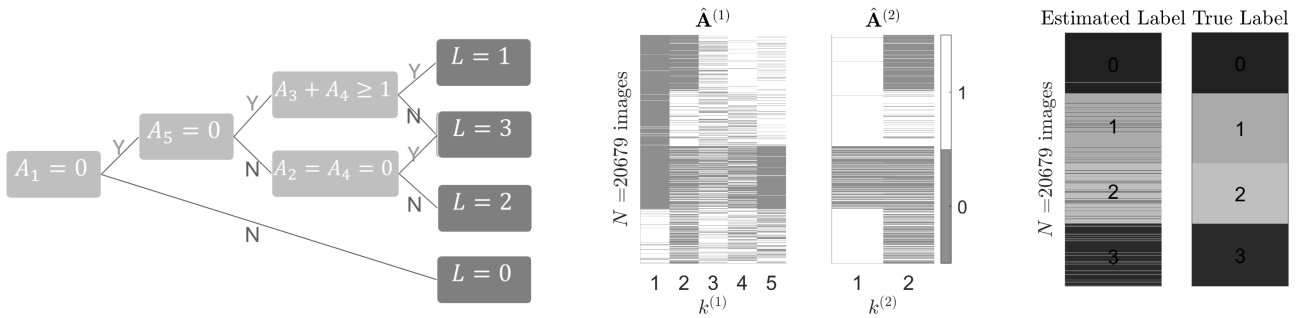


Figure 35.13: **Left:** Decision tree to estimate the digit $L = 0, 1, 2, 3$. **Centre:** Estimated latent representations $\hat{\mathbf{A}}^{(1)}, \hat{\mathbf{A}}^{(2)}$. **Right:** Estimated and true digits in the training set.

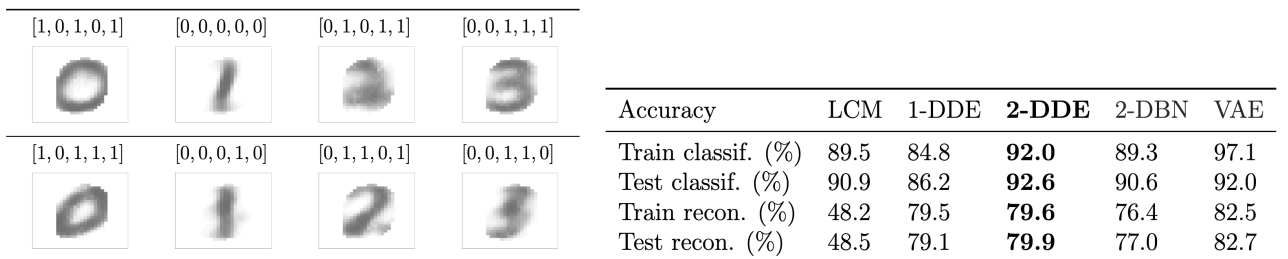


Figure 35.14: **Left:** Example images generated from DDE and their latent representations. **Right:** Classification accuracy and pixel-wise reconstruction accuracy for the MNIST dataset.

The learned shallower and deeper latent representations ($\hat{\mathbf{A}}^{(1)}, \hat{\mathbf{A}}^{(2)}$) are evaluated under two measures of performance: *classification accuracy* and *reconstruction accuracy*. Then, a decision tree that classifies the binary

latent representations into the categorical labels is built using misclassification error as the splitting criterion; see the left panel of Figure 35.13. The centre and right panels of Figure 35.13 display the estimated latent traits and labels. Our classifier leads to a high train/test accuracy of 92.0%/92.6%, even though our model is not fine-tuned for image classification. Although state-of-the-art machine-learning methods can achieve an accuracy as high as 97% [1198], the main goal here is interpretability and parsimony rather than classification; the DDE uses more limited information and is a less complex but more interpretable model that provides a generative process for the image. In the right panel of Figure 35.14, we compare the classification accuracy and pixel-wise reconstruction accuracy of the two-latent-layer DDE with alternative interpretable models (a latent class model (LCM) and a one-latent-layer DDE), as well as popular unsupervised machine-learning models (a two-latent-layer DBN and a variational autoencoder (VAE)). The results show that DDE performs well for both measures. We also display example images generated from DDE alongside their latent representations in the left panel of Figure 35.14, which illustrate various handwriting styles for each digit.

Count Data: Poisson-DDE Hierarchical Topic Modelling Next, we apply DDEs to learn hierarchical latent topics from text documents. In topic modelling, it is natural for the topics to be correlated with each other [960], and hierarchical topic modelling is often adopted [253, 672, 1283]. While many existing works assume a tree-structured hierarchy, DDEs flexibly allow multiple parents for each variable.

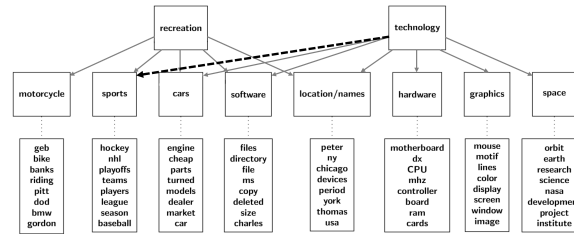


Figure 35.15: DDE estimated from the 20 Newsgroups dataset. For each shallow-layer latent variable, we display the top eight representative words. The width of each upper-layer arrow is proportional to the corresponding coefficient; the arrow from “technology” to “sports” indicates a negative value.

We analyse the 20 Newsgroups dataset [971]. After preprocessing and focusing on 12 newsgroups, the dataset consists of $N = 5,883$ documents and $J = 653$ words, representing a high-dimensional count-valued dataset. We fit the two-latent-layer DDE with a Poisson-distributed data layer (Poisson-DDE) $Y_j \mid (\mathbf{A}^{(1)} = \boldsymbol{\alpha}^{(1)}) \sim \text{Poi}(\exp[\beta_{j,0}^{(1)} + \sum_{k \in [K^{(1)}]} \beta_{j,k}^{(1)} \alpha_k^{(1)}])$ and latent dimensions $K^{(1)} = 8, K^{(2)} = 2$, and display the estimated latent structure in Figure 35.15. To better interpret individual latent variables, we define representative words for each topic k based on the discrepancy between $\beta_{j,k}$ and all other coefficients, $(\beta_{j,l})_{l \neq k}$. That is, for each index k , we choose the words j with the largest values of $\max\{\min\{\beta_{j,k} - \beta_{j,l} : l \neq k\}, 0\}$. We display the representative words for each latent variable in the bottom row of Figure 35.15. Here, each latent variable is named based on the representative words and the held-out newsgroup categories. Compared to the held-out tree structure of the 12 newsgroup labels, DDE discovers a more flexible DAG structure in Figure 35.15, allowing multiple parents for each topic to effectively model the complex label dependence. For example, ‘cars’, ‘software’, ‘location/names’ have both ‘recreation’ and ‘technology’ as parents, and many bottom-layer words are assigned to multiple topics.

Model	Latent dimensions	Train perplexity	Test perplexity	Neg. coherence	Similarity
LDA	8	499	512	276	43
LDA	256	269	515	321	(41450)
DPFA	8 – 2	289	499	211	5
DPFA	128 – 64	175	232	280	(3378)
DDE	8 – 2	322	398	270	0
DDE	8 – 3	322	399	275	0

Table 35.2: Train and test perplexity, negative coherence, and similarity scores of different models on the 20 Newsgroups dataset. For all measures, smaller values are better. We parenthesise similarity scores for models with different dimensions because the measure is not normalised.

We also compare the fit of DDE to other directed graphical models with matching latent dimensions: LDA [169] and DPFA-SBN [570]. LDA has a single latent layer with mixed membership scores, and DPFA-SBN is a

multilayer model with binary latent variables similar to DDEs. We consider the following three metrics widely used in topic modelling to measure different aspects of fit [281]: *perplexity*, *coherence*, and *similarity* (see definitions in Lee and Gu [1007]). Table 35.2 summarises the results and shows the promising fit of DDE. Compared to other models with the same latent dimensions, DDEs have better test perplexity and similarity.

Multimodal Educational Data: Bernoulli-Lognormal-DDE. We apply the DDE to an educational assessment dataset from the Trends in International Mathematics and Science Study (TIMSS) [523]. As the assessment is electronically conducted, multiple modalities of information are recorded. Here, we focus on two important modalities: *binary* response accuracy and *continuous* response time. For each individual student, our data consist of response accuracy (whether the student gave a correct answer) and response time (how long the student took) for each of the $J/2 = 29$ items. We use the same latent variables to model both data modalities and model the binary response accuracy via Bernoulli distributions, and model the continuous positive response times via Lognormal distributions: $Y_j \mid \boldsymbol{\alpha}^{(1)} \sim \text{lognormal}(\beta_{j,0}^{(1)} + \sum_{k \in [K^{(1)}]} \beta_{j,k}^{(1)} \alpha_k^{(1)}, \gamma_j)$.

Response \ Latent skill	$A_1^{(2)}$	$A_1^{(1)}$: Number	$A_2^{(1)}$: Algebra	$A_3^{(1)}$: Geometry	$A_4^{(1)}$: Data and Prob
Agree a lot	0.62	0.59	0.62	0.42	0.57
Agree a little	0.50	0.47	0.52	0.34	0.43
Disagree a little	0.37	0.33	0.38	0.31	0.37
Disagree a lot	0.29	0.29	0.32	0.22	0.29

Table 35.3: Average latent variable estimates for each response category for the question “*Mathematics is one of my favorite subjects*”. The table shows the higher-order latent variable and four representative first-layer skills.

We fit the two-latent-layer DDE with $K^{(1)} = 7$, $K^{(2)} = 1$, and estimate each student’s latent skills by the latent attribute profile with the largest posterior probability. We compare the estimated latent variables with the held-out information of each student’s categorical response to a survey question: “*Mathematics is one of my favorite subjects*”, and display the results in Table 35.3. The first column shows that the higher-order latent variable, $A_1^{(2)}$, is highly correlated with the extent to which students like mathematics. This suggests that $A_1^{(2)}$ can be interpreted as a general indicator of the students’ interest in mathematics, while the fine-grained latent variables $\mathbf{A}^{(1)}$ represent students’ specific skill mastery profiles. In addition, we observe that the students who enjoy math tend to have a higher probability of mastering specific skills as well. This is coherent with the fact that the estimated $\mathbf{B}^{(1)}$ -coefficients are nonnegative for both modalities.

35.6 Concluding Remarks

In summary, identifiability provides the theoretical foundation for interpretable and reproducible Bayesian deep learning. This chapter unified recently proposed identifiable deep generative frameworks and outlined their key theoretical guarantees. By embedding algebraic identifiability conditions as graphical constraints within hierarchical Bayesian modelling, these approaches demonstrate that deep representation learning can be both *statistically sound and practically interpretable*.

Chapter 36

Bayesian mixture of experts in BDL

Mixture-of-experts (MoE) models, originally proposed by Jacobs et al. [828], extend traditional mixture models in the context of regression, classification, and beyond. In the context of modern deep learning, there has been a recent resurgence in interest in such models, as they show promise in scaling computation while retaining model expressiveness. The aim of this chapter is to study a particular version of MoE models, those featuring *softmax gating*, which has been popularized in deep learning applications and which naturally lends itself to a treatment in the Bayesian learning framework.

36.1 Introduction

At a high level, MoE models aggregate multiple specialized submodels, known as *experts*, and employ a learnable *gating function* or *router* to determine the weight of (or probability of assignment to) each expert [654]. In contrast to traditional mixture models with static mixing proportions, the gating function in MoE models flexibly assigns input-dependent weights to experts, in such a way that experts that are more relevant to the input receive larger weights than others. As a result, MoE models are more expressive and adaptive than mixture models when one wishes to model the conditional distribution of an output given inputs. Specifically, MoE can lead to more accurate estimation than conventional mixture models in regression/classification settings where the conditional distribution of the response Y given the predictor vector X varies significantly as a function of X .

The basic MoE paradigm has been extended in several directions. Jordan and Jacobs [850] devised hierarchical mixtures-of-experts (HME), allowing experts to be organized in a tree structure for better expert specialization. Recently, authors in [1508] introduced a sparse version of MoE in which only a subset of experts is activated per input during training. This conditional-computation strategy helps scale up the model capacity significantly without a proportional increase in computation. Thanks to their scaling power, there has been a surge of interest in using sparse MoE models in a wide range of fields in deep learning, including language modelling [392, 843], computer vision [1398], multi-task learning [717], multi-modal learning [704], domain adaptation [1022, 1242], continual learning [989] and parameter-efficient fine-tuning [425, 990]. Despite their success, these implementations rely on maximum likelihood estimation, often via the Expectation-Maximization algorithm, which exposes them to issues of overfitting and fixed model complexity [1143].

If uncertainty quantification is desired, adapting the Bayesian paradigm to MoE models is a sensible choice. Authors in [1313] developed a fully Bayesian MoE model using MCMC methods, which allowed inference over both expert parameters and router parameters. Simultaneously, authors in [1728] proposed a Bayesian framework for inferring the parameters of an MoE model based on ensemble learning by variational free energy minimization, enabling more scalable inference by optimizing a lower bound on the marginal likelihood. These contributions marked a shift from point estimates to full posterior distributions, improving generalization and interpretability. Bayesian MoE models bring together the strengths of modular learning and Bayesian probabilistic inference: by decomposing complex prediction tasks into simpler subtasks solved by specialized experts and coordinated through a gating mechanism, MoE models provide flexibility and interpretability; the Bayesian formulation further enhances this framework by addressing overfitting, model selection, and uncertainty quantification, combining the benefits of both paradigms.

A key theoretical advancement in the development of Bayesian MoE models came with the work of Ueda and Ghahramani [1645], who proposed a variational Bayes framework capable of performing automatic model selection. Unlike traditional MoE models that require the number of experts to be specified a priori, their method treated this quantity as a latent random variable, enabling the model to infer an appropriate level of complexity

from data. By optimizing an evidence lower-bound (ELBO), their approach facilitated the adaptive pruning of redundant or unnecessary experts, thus balancing model fit and parsimony. Building on this foundation, authors in [1381] introduced a nonparametric Bayesian formulation of MoEs by incorporating Dirichlet Processes (DPs) for model selection and Gaussian Processes (GPs) as flexible expert functions. This allows the number of mixture components to grow dynamically with the complexity of the data, removing the need to fix the number of experts in advance. Moreover, by employing GPs, the model can capture highly nonlinear relationships within each expert domain while retaining a coherent probabilistic framework.

Inference in Bayesian MoE models, however, remains a non-trivial task due to the complexity of the posterior distribution, particularly when dealing with non-conjugate priors or hierarchical model structures. Traditional Markov Chain Monte Carlo (MCMC) methods, such as those discussed in [1313], provide asymptotically exact samples from the posterior, but they are often computationally expensive and slow to converge, especially in high-dimensional or large-scale settings. As an alternative, variational inference techniques have been developed to approximate the posterior more efficiently. Notably, Bishop and Svensén [162] applied variational methods to MoEs, demonstrating that such approximations could yield accurate predictions with significantly reduced computational costs. Variational inference also provides a natural framework for model comparison and selection via the ELBO, further enhancing its utility. In recent years, the rise of probabilistic programming languages such as Stan [240] and Pyro [158] has dramatically simplified the implementation of Bayesian MoEs. These frameworks support advanced inference algorithms, including Hamiltonian Monte Carlo (HMC), No-U-Turn Sampler (NUTS) [762], and stochastic variational inference [763], which enable more scalable and expressive models for the expert functions and the gating functions. The modular design of probabilistic programming also allows seamless integration with deep learning architectures, facilitating the construction of hybrid models that leverage both probabilistic reasoning and the representational power of neural networks for MoE.

Bayesian MoE models have demonstrated their utility across a broad range of application domains. In time-series forecasting, they offer robustness to regime shifts and non-stationary patterns by allowing expert specialization over different temporal segments [1728]. In personalized medicine, Bayesian MoEs have been used to capture patient heterogeneity, account for latent subpopulations, and improve treatment effect estimation in heterogeneous clinical cohorts [1784]. In control and robotics, they enable interpretable mode switching in dynamical systems, allowing agents to transition smoothly between operational modes in response to changing environmental conditions [81]. Despite the growing dominance of non-Bayesian deep learning methods, recent work at the intersection of Bayesian inference and large-scale neural architectures is reviving interest in probabilistic MoEs.

36.2 Softmax-gated mixture of experts models

In this section, we introduce simple softmax MoE models for regression and classification, and provide intuition on the generative mechanism they imply.

36.2.1 Regression

Consider an iid sample $(Y_i, X_i)_{i=1}^n$, where each observation i consists of a response variable $Y_i \in \mathbb{R}$ and an input vector $X_i \in [0, 1]^d$. As in classical regression tasks, the goal is to learn the conditional distribution of the response given the inputs. To that end, we assume that the sample is generated as follows: $(Y_i, X_i) \stackrel{\text{iid}}{\sim} g_{G^*}$, where the joint distribution g_{G^*} factorizes as

$$g_{G^*}(Y, X) = f_{G^*}(Y | X) \times p(X).$$

Here, because we are not interested in learning the marginal distribution of the input vector, we simply assume p to be known and to coincide with the uniform density on $[0, 1]^d$. As for the conditional density $f_{G^*}(\cdot | X)$, setting

$$G^* := \sum_{j=1}^{K^*} \exp(\alpha_{0j}^*) \delta_{(\alpha_{1j}^*, \beta_j^*, \sigma_j^{*2})},$$

we assume the form

$$f_{G^*}(Y | X) := \sum_{j=1}^{K^*} \frac{\exp(\alpha_{0j}^* + X^\top \alpha_{1j}^*)}{\sum_{\ell=1}^{K^*} \exp(\alpha_{0\ell}^* + X^\top \alpha_{1\ell}^*)} \mathcal{N}(Y | \zeta(X, \beta_j^*), \sigma_j^{*2}). \quad (36.1)$$

where $\alpha_{0j}^* \in \mathbb{R}$, $\alpha_{1j}^* \in \mathbb{R}^d$, $\zeta : [0, 1]^d \times \mathbb{R}^E \rightarrow \mathbb{R}$ is a mean (or *expert*) function parametrised by $\beta \in \mathbb{R}^E$, and $\mathcal{N}(\cdot | \mu, \sigma^2)$ stands for the normal density with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$. A convenient way to parametrise

this conditional density is through the measure¹ G^* defined above, which collects all of the model parameters and which, as we will see later in the chapter, may be a useful object for the study of inferential procedures.

Let us now interpret the structure of this Softmax-gated Mixture-of-Gaussian-Experts (SMoGE) generative model. Given a value $X \sim p$ of the inputs, the response Y is generated according to a convex combination of K^* normal distributions, termed *experts*, each with its own input-dependent mean $\zeta(X, \beta_j^*)$ and variance σ_j^{*2} . Here the expert function ζ can be as simple as an affine transformation of X (so β is a linear coefficient vector together with a bias term), or a more flexible mapping based, for instance, on a deep neural network (with β collecting the network weights).

An important feature, which sets apart the MoE model from traditional mixtures, is that the weight assigned to each expert depends on the input value X (rather than being a constant). In particular, this dependence is modelled with the softmax function

$$S : \mathbb{R}^{K^*} \rightarrow \Delta_{K^*}$$

$$z \mapsto \frac{\exp(z)}{\mathbf{1}^\top \exp(z)},$$

where $\Delta_{K^*} := \{\mathbf{w} \in \mathbb{R}^{K^*} : \mathbf{w} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{w} = 1\}$ is the $(K^* - 1)$ -dimensional simplex, and the function $\exp$ is applied element-wise to each coordinate of $\mathbf{z} \in \mathbb{R}^{K^*}$. In this case, the softmax function takes as input the vector of linear combinations² $(\alpha_{0j}^* + X^\top \alpha_{1j}^*)_{j=1}^{K^*}$ and outputs the weights corresponding to the K^* experts.

To further understand the structure of the model, it is useful to introduce a set of latent *gating* variables $Z_1, \dots, Z_n \in \{1, \dots, K^*\}$ and rewrite the model in hierarchical form as follows:

$$Y_i \mid X_i, Z_i \stackrel{\text{ind}}{\sim} \mathcal{N}(\cdot \mid \zeta(X_i, \beta_{Z_i}^*), \sigma_{Z_i}^{*2}),$$

$$Z_i \mid X_i \stackrel{\text{ind}}{\sim} \mathcal{C}\left(\cdot \mid (1, \dots, K^*), S\left((\alpha_{0j}^* + X_i^\top \alpha_{1j}^*)_{j=1}^{K^*}\right)\right),$$

where $Z \sim \mathcal{C}(\cdot \mid \mathbf{a}, \mathbf{w})$, for $\mathbf{a} \in \mathbb{R}^{K^*}$ and $\mathbf{w} \in \Delta_{K^*}$, means $Z = a_j$ with probability w_j for all $j = 1, \dots, K^*$.

In other words, in the SMoGE model the conditional generation of the response variable Y_i can be interpreted as follows: given a certain input value $X_i \sim p$, first an indicator Z_i is generated, according to X_i -dependent softmax probabilities, which “gates” observation i to expert $Z_i \in \{1, \dots, K^*\}$. Then, given this assignment, the output Y_i is generated by a normal distribution with X_i -dependent mean parametrised by ζ . This generative view further clarifies a key source of added flexibility of SMoGE models compared to simple mixtures: SMoGE models allow for output observations whose distribution depends on inputs not only in terms of the mean structure, but also in terms of the probability of being assigned to a specific mean structure itself.

36.2.2 Classification

If the output belongs to a finite set of classes $Y_i \in \{1, \dots, C\}$ (instead of being continuous as in a regression problem), then we face a classification problem—say, to determine whether a certain image X_i depicts a certain type of animal Y_i among a list labelled 1 through C [1239]. Similarly to before, this can be tackled by specifying the following MoE generative model expressed through the conditional probability mass function of Y given $X \sim p$:

$$f_{G^*}(Y \mid X) :=$$

$$\sum_{j=1}^{K^*} \frac{\exp(\alpha_{0j}^* + X^\top \alpha_{1j}^*)}{\sum_{\ell=1}^{K^*} \exp(\alpha_{0\ell}^* + X^\top \alpha_{1\ell}^*)} \times \mathcal{C}(Y \mid \{1, \dots, C\}, S(\zeta(X, \beta_{j1}^*), \dots, \zeta(X, \beta_{jC}^*))).$$

The implied generative model is analogous to the regression case, and can be similarly rewritten in hierarchical form by introducing suitable latent gating variables. The only difference is that, once an observation has been gated to expert j according to input-dependent probabilities, the output is generated according to an expert-specific categorical distribution on $\{1, \dots, C\}$, with probabilities parametrised by a softmax transformation of the expert functions $\zeta(X, \beta_{j1}^*), \dots, \zeta(X, \beta_{jC}^*)$. Please take a moment to notice that, while the softmax function appears twice in this model specification, it serves two distinct purposes in each case: in the first instance, to gate observations to a certain expert; in the second one, to assign each observation to one of the C classes under consideration.

¹Please note that G^* is not a probability measure, as its weights, while positive, are not guaranteed to sum to 1.

²Please note that we force the softmax gating to depend linearly on the input vector for a matter of convenience. This linear specification can readily be replaced by more flexible functions, if needed.

36.3 Fitting MoE models

Having defined the basic MoE models, we now turn to the question of how to fit them in practice using Bayesian methods.

A first basic issue to deal with is the fact that the number K^* of experts is not in general known in advance. To address this, there are two basic strategies at the disposal. First, we can simply specify an MoE model with $K \in \mathbb{N}$ experts, where K is large enough so that $K \geq K^*$. In this case, we talk about an *overspecified* MoE model. Alternatively, as we will be working in the Bayesian framework, one can treat the number of experts K as yet another parameter to estimate with the available data by placing a prior on it (say, a shifted Poisson distribution with support on the positive integers) and making posterior inference on it together with the other parameters. Each strategy has its advantages and disadvantages. Overspecified models are, on the one hand, relatively simple to fit (see below) and are capable of approximating the true model—just think of a sequence of MoE density with gating parameters that make the gating weights of the extra experts closer and closer to 0; on the other hand, in real world situations where data is limited and careful modelling is essential for good inference and prediction, specifying too high a value of K may result in poor performance. Instead, placing a prior for K leads to principled inference on this unknown quantity, and allows to learn it automatically with the other parameters. However, notice that, as K varies, so does the dimension of the parameter space, as more (or fewer) gating and expert parameters are included in the model. This complicates significantly the computational burden of fitting the model, as ad hoc procedures that take into account the *transdimensional* nature of the inference problem are needed [662]. To keep the exposition clear, we focus on the first choice, i.e., we fix K and we assume that it has been selected large enough so as to subsume the true underlying MoE model.³

Given the model⁴ we specified in the previous section, we are interested in performing inference on two distinct sets of parameters: the gating parameters $(\alpha_{0j}, \alpha_{1j})_{j=1}^K$, and the expert parameters $(\beta_j)_{j=1}^K$. According to the standard Bayesian pipeline, then, we assign a prior distribution Π to these parameters. To keep the notation compact, recall that the measure $G = \sum_{j=1}^K \exp(\alpha_{0j}) \delta_{(\alpha_{1j}, \beta_j, \sigma_j^2)}$ contains all the information about the MoE parameters, so we will think of the prior as being placed on the space of all such measures G . While it may seem abstract, this is just to avoid notational clutter, and all we mean by $\Pi(G)$ is that a prior is placed on the finite-dimensional (Euclidean) space of MoE model parameters.

To make inference, we compute the posterior distribution

$$\Pi(G \mid Y_{1:n}, X_{1:n}) \propto \Pi(G) \times \prod_{i=1}^n f_G(Y_i \mid X_i). \quad (36.2)$$

If we care about predicting a new value Y_{n+1} of the output given a certain input value X_{n+1} , we may also compute the posterior predictive distribution

$$\hat{f}_n(Y_{n+1} \mid X_{n+1}) := \int f_G(Y_{n+1} \mid X_{n+1}) \Pi(dG \mid Y_{1:n}, X_{1:n}). \quad (36.3)$$

It should be clear that, given the rather complicated density $f_G(Y \mid X)$ associated to MoE models, it is not possible to compute exactly the posterior distribution (36.2) nor the predictive (36.3). Therefore, we now briefly examine two popular options to perform computation: Markov Chain Monte Carlo (MCMC) sampling and variational inference (VI).

36.3.1 Markov Chain Monte Carlo sampling

A first popular strategy is to use MCMC sampling to approximate the posterior distribution. As discussed in depth in other parts of the book, this involves simulating a Markov chain having as state space the parameter space and as a stationary distribution the posterior distribution, and then treating the realizations of the chain (after discarding an appropriate burn-in initial segment to ensure convergence of the chain to stationarity) as samples from the posterior.

³Two further subtle points, related to the problem of selecting K , are the following: first, while a true data-generating process may exist, the latter may not be expressible as an MoE model, so a true K^* may not exist, and one may rather want to select a value of K that ensures good predictive performance; second, if one takes the subjective Bayesian perspective seriously, no “true” data-generating process—in the usual iid sense—may exist at all (let alone a true number of experts). Therefore, although convenient to study the theoretical properties of the models, the implications of choosing a “correct” value of K^* should not be over-interpreted.

⁴For the sake of convenience, we focus on MoE models for regression, although the treatment easily extends to the classification setting as well.

The smooth parametrisation of MoE models lends them to any state of the art MCMC sampler, such as Hamiltonian Monte Carlo and Langevin Dynamics. We therefore omit a detailed treatment of this aspect, and refer the readers to the relevant chapters within the book. Nevertheless, a couple of observations, which are specific to MoE models, are in order.

First, the similarity with traditional mixture models may lead one to expect to have access to full conditional distributions that would allow to perform Gibbs updates for the gating and expert parameters. This is unfortunately not the case in general, at least for two reasons:

1. The softmax gating parametrisation, while intuitive and very flexible, does not allow for conditionally conjugate priors, especially as the input-dependent weights interact with the expert densities when one wishes to perform MCMC updates;
2. The expert function ζ is rarely of simple enough form (e.g., linear) so as to allow Gibbs updates.

Nevertheless, in simple cases such as in the in the context of linear regression with conditionally conjugate priors, it may be possible to perform Gibbs updates for at least some of the parameters.

Second, MoE models naturally lead to a model-based clustering of the observations. Recall that, by introducing the latent expert assignment variables Z_i , one can interpret the generative model as first assigning observations to a certain cluster (expert), then generating a response according to a conditional density that is specific to the cluster. Therefore, by making inference on the latent Z_i 's, one can learn useful features of the clustering structure of the observed sample. It turns out that many useful MCMC algorithms do rely on this augmentation strategy and, as a consequence, allow to obtain posterior summaries of the data clustering.

36.3.2 Variational inference

A second approach, which is standard in the BDL literature, is to fit MoE models via variational inference. The basic idea behind this strategy is simple: instead of sampling from the posterior distribution (asymptotically) exactly, find the distribution that best approximates the posterior within a tractable family of distribution (the so called variational family). This methodology is also standard and explored in depth elsewhere in the book, therefore we omit a detailed treatment of the topic.

The following, however, are two important remarks. First, variational inference is to be preferred, in terms of scalability, to MCMC, especially in situations where the expert functions are highly non-linear (for instance, deep neural networks) and where the parameter space is very high-dimensional: in these cases, Metropolis-Hastings adjustment will be most likely needed, a feature that, paired with high-dimensionality, is very likely to lead to a high computational cost of MCMC. As usual, however, depending on the chosen variational family and the nature of the data, variational inference may suffer from lower posterior approximation accuracy compared to MCMC (which is asymptotically exact), especially in terms of posterior variance under-estimation. Therefore, if accurate posterior uncertainty quantification is essential, MCMC may be preferable even at the expense of higher computational costs.

Second, the complicated likelihood associated to MoE models makes it infeasible to employ simple variational inference algorithms based on conjugate priors (e.g., coordinate ascent variational inference). Instead, more flexible methods are necessary, such as *black-box variational inference* [939, 1377]. The latter is a powerful technique that, by approximating the gradient of the evidence lower-bound (the objective we optimize in variational inference) with an average of terms with respect to Monte Carlo samples from the variational family, allows to fit elaborate models like the MoE ones. Key strengths of this strategy are that (i) it allows to take advantage of the powerful automatic differentiation tools available within routinely used modern deep learning libraries, and (ii) it easily allows for data mini-batching, which is essential if n is large.

36.4 Asymptotic properties

In this section, we study some asymptotic consistency properties of the softmax-gated Gaussian mixture of experts, namely the convergence behavior of density estimation and parameter estimation. To give some preliminary background, asymptotic consistency is a key criterion for evaluating statistical estimation procedures, and its study in the context of Bayesian posterior distributions has a long history. Heuristically, one concludes that the posterior distribution in a given model is consistent if the mass it assigns to any neighbourhood (no matter how small) of the data-generating parameter converges to 1 almost surely (or in probability) as the sample size grows to infinity. The first results in this direction trace back to Doob [442] establishing, under mild conditions, posterior consistency on a set of data-generating parameters of prior measure 1. Later developments focused on

parametric models for iid data, such as Walker [1695] and Berk [126], building on frequentist consistency results for maximum likelihood estimators (however see Barileto et al. [96] for a recent perspective on parametric consistency independent of frequentist techniques). The focus has since mostly shifted towards consistency in nonparametric models [100, 616, 1476, 1696].

In what follows, instead of looking at consistency, we look at the related concept of contraction rates: rather than making sure that the MoE posterior distribution consistently estimates the true MoE density and parameters by placing vanishing mass outside of any neighbourhood of the true object of inference, we ask *how fast* we can shrink such neighbourhoods and still obtain posterior concentration within them. To simplify the exposition, we will only focus on regression, although quite similar results can be stated in the context of classification.

36.4.1 Density estimation

As described in Section 36.2.1, a primary objective in classical regression tasks is to learn the conditional distribution of the response given the input, which is characterized by its conditional density function. Having access to this knowledge allows us to capture the convergence rate of density estimation, expressed as a function of the sample size. As a consequence, we are able to determine the amount of data necessary to approximate a softmax-gated MoE density within a given estimation error. In particular, let us recall that the iid sample of size n , $(Y_i, X_i)_{i=1}^n$, is assumed to have been generated from the joint distribution with the density function $g_{G^*}(Y, X) = f_{G^*}(Y | X) \times p(X)$, where the conditional density $f_{G^*}(\cdot | X)$ is given by

$$f_{G^*}(Y | X) := \sum_{j=1}^{K^*} \frac{\exp(\alpha_{0j}^* + X^\top \alpha_{1j}^*)}{\sum_{\ell=1}^{K^*} \exp(\alpha_{0\ell}^* + X^\top \alpha_{1\ell}^*)} \cdot \mathcal{N}(Y | \zeta(X, \beta_j^*), \sigma_j^{*2}). \quad (36.4)$$

A question arising here is how accurately we can estimate the joint density function $g_{G^*}(Y, X)$ using the iid sample $(Y_i, X_i)_{i=1}^n$. To answer this question, it is necessary to choose an appropriate metric to measure the distance between the estimated and the ground-truth density among the following three popular choices:

(i) *Hellinger distance*. For any two probability density functions p and q defined on a Borel set $S \subseteq \mathbb{R}^m$, the Hellinger distance is defined as

$$d_H(p, q) := \left(\frac{1}{2} \int_S (\sqrt{p} - \sqrt{q})^2 d\lambda \right)^{1/2},$$

where λ denotes a σ -finite measure on S ;

(ii) *Total Variation (TV) distance*. The TV distance is given by

$$d_{TV}(p, q) := \int_S |p - q| d\lambda;$$

(iii) *Kullback-Leibler (KL) divergence*. The KL divergence is computed as

$$D_{KL}(p||q) := \int_S \log(p/q) p d\lambda.$$

In the Bayesian asymptotics literature, the TV distance and Hellinger distance are usually preferred over the KL divergence because they provide more stable and decision-theoretically meaningful measures of density discrepancies (though they are topologically weaker than the KL divergence). In particular, the KL divergence can be infinite when estimated densities do not admit the same support as the ground-truth density. By contrast, both the TV and Hellinger distances remain finite and bounded, making them more robust to support mismatches and better behaved for asymptotic analysis. Because the Hellinger and TV metrics are well known to be topologically equivalent, we will measure density convergence in the Hellinger distance without loss of generality. Nevertheless, the KL divergence will be employed to measure prior concentration around the true density function.

Before presenting the main result on posterior contraction rates for density estimation, we need to introduce some mild conditions on the parameter space. In particular, we will work under the assumption that all K experts share the same parameter space Θ , i.e. $\theta = (\alpha_{0j}, \alpha_{1j}, \beta_j, \sigma_j^2) \in \Theta$, and we will denote by H the subspace of expert parameters $\eta = (\beta_j, \sigma_j^2) \in H$. For any $K \in \mathbb{N}$, denote by $\mathcal{G}_K$ the set of all K -expert mixing measures, that is,

$$\mathcal{G}_K := \left\{ G = \sum_{j=1}^K \exp(\alpha_{0j}) \delta_{(\alpha_{1j}, \beta_j, \sigma_j^2)} : (\alpha_{0j}, \alpha_{1j}, \beta_j, \sigma_j^2) \in \Theta, j = 1, \dots, K \right\}.$$

Assumption 36.1. *The parameter space Θ is compact. In particular, the variance space is bounded away from 0.*

Assumption 36.2. *The prior Π satisfies*

$$\Pi \left(D_{\text{KL}}(g_{G^*} \| g_G) \leq A \frac{\log t}{t} \right) \geq \frac{B}{t} \quad (36.5)$$

for some $A, B > 0$ and all large $t > 0$.

Assumption 36.3. *There exists $L > 0$ such that, for all $x \in \mathcal{X}$ and $\beta, \beta' \in \mathbb{R}^E$ in the parameter space,*

$$|\zeta(x, \beta) - \zeta(x, \beta')| \leq L \|\beta - \beta'\|. \quad (36.6)$$

Assumption 36.1 is a standard compactness assumption that simplifies theoretical analysis, while 36.2 requires that enough prior mass be placed around the data-generating parameters [619]; see also Barileto et al. [97] for more details on the second requirement for MoE models. Additionally, Assumption 36.3 allows the expert function to be Lipschitz continuous. Given these three assumptions, we are ready to state main results about the convergence rate of density estimation.

Theorem 36.1. *Consider a SMOGE model. Under Assumptions 36.1-36.3, the following holds for any sequence $M_n \rightarrow \infty$:*

$$\lim_{n \rightarrow +\infty} \Pi \left(\left\{ G \in \mathcal{G}_K : d_H(g_G, g_{G^*}) \gtrsim M_n \sqrt{\log(n)/n} \right\} \mid (X_i, Y_i)_{i=1}^n \right) = 0$$

in $g_{G^*}^n$ -probability, where $\gtrsim$ denotes an inequality up to a constant.

From Theorem 36.1, we learn that under the Hellinger distance, the posterior distribution over MoE densities contracts around the ground-truth density function at a rate of order $\mathcal{O}_P(\sqrt{\log(n)/n})$, which is parametric in the sample size n .

Now, it is natural to ask about the consistency of the above posterior convergence on the space of densities, or equivalently, the identifiability of softmax-gated Gaussian MoE (SMoGE). In particular, assume that there are two conditional density functions corresponding to two softmax-gated Gaussian MoE models that admit the same values almost everywhere. Then, in Proposition 36.1, we will determine if these two models share the same parameter set. For that purpose, it is necessary to introduce an identifiability assumption on the expert function and a distinction assumption on the model parameters.

Assumption 36.4. *The expert function $\zeta : [0, 1]^d \times \mathbb{R}^E \rightarrow \mathbb{R}$ is identifiable, that is, if for each $K \in \mathbb{N}$, the following holds: if there exist distinct parameters $(\beta_1, \beta_2, \dots, \beta_K)$ and $(\beta'_1, \beta'_2, \dots, \beta'_K)$ such that for almost every $x \in \mathcal{X}$, we can find permutation function $\sigma_x : \{1, 2, \dots, K\} \rightarrow \{1, 2, \dots, K\}$ such that $\zeta(x, \beta_{\sigma_x(i)}) = \zeta(x, \beta'_i)$ for all $1 \leq i \leq K$, then $\{\beta_1, \beta_2, \dots, \beta_K\} \equiv \{\beta'_1, \beta'_2, \dots, \beta'_K\}$.*

For example, it can be validated that for any injective function $\varphi : \mathbb{R} \rightarrow \mathbb{R}$, an expert function of the form $\zeta(x, (\beta_1, \beta_0)) = \varphi(\beta_1^\top x + \beta_0)$, where $\beta_1 \in \mathbb{R}^d$ and $\beta_0 \in \mathbb{R}$, is identifiable.

Assumption 36.5. *The discrepancies between gating parameters $\alpha_{1i}^* - \alpha_{1j}^*$, for $1 \leq i < j \leq K^*$, are pairwise distinct. In addition, expert parameters $(\beta_j^*, \sigma_j^{*2})$, for $1 \leq j \leq K^*$, are also pairwise different. The same holds for the parameter configuration associated to Π -almost every mixing measure G .*

Proposition 36.1. *Under Assumptions 36.4-36.5, if there exist two mixing measures $G = \sum_{j=1}^K \exp(\alpha_{0j}) \delta_{(\alpha_{1j}, \beta_j, \sigma_j^2)}$ and $G^* = \sum_{j=1}^{K^*} \exp(\alpha_{0j}^*) \delta_{(\alpha_{1j}^*, \beta_j^*, \sigma_j^{*2})}$ on Θ such that $g_G(y, x) = g_{G^*}(y, x)$ for almost every (y, x) , then we obtain $K = K^*$ and*

$$G \equiv G_{t_0, t_1}^* := \sum_{j=1}^{K^*} \exp(\alpha_{0j}^* + t_0) \delta_{(\alpha_{1j}^* + t_1, \beta_j^*, \sigma_j^{*2})},$$

for some $t_0 \in \mathbb{R}$ and $t_1 \in \mathbb{R}^d$.

The result of Proposition 36.1 indicates that the softmax-gated Gaussian MoE is identifiable up to the translation of gating parameters. This dependence on the gating parameter translation comes from the softmax gating structure in equation (36.4). More specifically, the values of softmax weights will remain unchanged if we translate the gating parameters α_{0j}^* and α_{1j}^* to $\alpha_{0j}^* + t_0$ and $\alpha_{1j}^* + t_1$, respectively, for some $t_0 \in \mathbb{R}$ and $t_1 \in \mathbb{R}^d$, which is mathematically presented as

$$\frac{\exp(\alpha_{0j}^* + t_0 + X^\top(\alpha_{1j}^* + t_1))}{\sum_{\ell=1}^{K^*} \exp(\alpha_{0\ell}^* + t_0 + X^\top(\alpha_{1\ell}^* + t_1))} = \frac{\exp(\alpha_{0j}^* + X^\top \alpha_{1j}^*)}{\sum_{\ell=1}^{K^*} \exp(\alpha_{0\ell}^* + X^\top \alpha_{1\ell}^*)}, \quad (36.7)$$

for all $j = 1, 2, \dots, K^*$. It should be noted that if we fix a pair of gating parameters to be zero, e.g., $\alpha_{0K^*}^* = 0$ and $\alpha_{1K^*}^* = 0_d$, then the numerator of the last softmax weight will be equal to one, that is, $\exp(\alpha_{0K^*}^* + X^\top \alpha_{1K^*}^*) = 1$. Therefore, the softmax weights will no longer be invariant to the translation of their parameters. As a consequence, the softmax-gated Gaussian MoE will become identifiable. Then, we can say that the posterior convergence on the space of densities in Theorem 36.1 is consistent.

36.4.2 Parameter estimation

In this section, we examine the convergence properties of parameter estimation within the SMoGE framework. Recall that during the training of MoE architectures in deep learning, each expert is expected to specialize in distinct subproblems or latent regions of the data distribution. For example, in large-scale language models based on the MoE paradigm [392, 603, 843], certain experts will capture syntax, another might focus on semantics, while the rest concentrate on context.

From a Bayesian perspective, this specialization can be viewed as a form of posterior concentration, where each expert's parameters adapt to a particular mode of the underlying data generation process. The rate at which this concentration occurs—i.e., the rate of expert specialization—is therefore a key quantity of interest [356, 1241, 1257]. A natural approach to studying this phenomenon is through the convergence behavior of parameter estimation, as the asymptotic or finite-sample convergence rates of the model parameters directly govern the rate at which experts specialize.

Intuitively, a faster convergence rate indicates that an expert requires fewer data samples to effectively capture its designated region of the input space, given a desired approximation accuracy. By contrast, slower convergence suggests higher sample complexity and delayed specialization. Consequently, analyzing the convergence of parameter estimation not only provides theoretical insights into the dynamics of expert learning but also informs strategies to improve the sample efficiency and inductive bias of MoE models.

Notably, the posterior convergence on the space of densities in Theorem 36.1 plays a vital role in establishing the convergence rate of parameter estimation. In particular, if there exists a metric to capture the discrepancies between parameter estimation and ground-truth parameters denoted by $\mathcal{L}(G, G^*)$ such that it is lower bounded by the Hellinger distance between density estimation and the ground-truth density, that is,

$$d_H(g_G, g_{G^*}) \gtrsim \mathcal{L}(G, G^*), \quad (36.8)$$

for any feasible mixing measures G . Then, by combining the results of Theorem 36.1 and equation (36.8), we will obtain the convergence rate of parameter estimation. Therefore, in the sequel, we will focus on constructing the metric $\mathcal{L}(G, G^*)$. To this end, we divide the analysis into two main settings referred to as (i) *exactly-specified setting* where the number of ground-truth experts K^* is known, and (ii) *over-specified setting* where K^* becomes unknown.

Exactly-specified setting. First, we perform the convergence analysis of parameter estimation when the number of ground-truth experts K^* is known. For simplicity, we set the last pair of gating parameters to zero to ensure that the softmax-gated Gaussian MoE is identifiable. Next, it should be noted that the mixing measure $G^* = \sum_{j=1}^{K^*} \exp(\alpha_{0j}^*) \delta_{(\alpha_{1j}^*, \beta_j^*, \sigma_j^{*2})}$ is only identifiable up to a permutation of parameters. In particular, if there exist two equivalent mixing measures, $G \equiv G'$, then it only implies $(\alpha_{0j}, \alpha_{1j}, \beta_j, \sigma_j^2) = (\alpha_{0\phi(j)}^*, \alpha_{1\phi(j)}^*, \beta_{\phi(j)}^*, \sigma_{\phi(j)}^{*2})$ for some permutation ϕ of the set $\{1, 2, \dots, K^*\}$. To identify that permutation, we employ the notion of Voronoi cells, which is defined as follows. Given an arbitrary mixing measure G with K^* components, we partition its components into the following Voronoi cells, which are generated by the components of ground-truth mixing measure G^* :

$$\mathcal{V}_j \equiv \mathcal{V}_j(G) := \{i \in \{1, 2, \dots, K^*\} : \|\omega_i - \omega_j^*\| \leq \|\omega_i - \omega_\ell^*\|, \forall \ell \neq j\}, \quad (36.9)$$

where $\omega_i := (\alpha_{1i}, \beta_i, \sigma_i^2)$ and $\omega_j^* := (\alpha_{1j}^*, \beta_j^*, \sigma_j^{*2})$ for any $j \in \{1, 2, \dots, K^*\}$. Notably, when the components of G are sufficiently close to those of G^* , then each of these Voronoi cells will contain one element. As a result,

we can characterize the aforementioned permutation of parameters. Then, the corresponding Voronoi-based loss capturing parameter discrepancies is given by

$$\begin{aligned} \mathcal{L}(G, G^*) &:= \sum_{j=1}^{K^*} \left| \sum_{i \in \mathcal{V}_j} \exp(\alpha_{0i}) - \exp(\alpha_{0j}^*) \right| \\ &\quad + \sum_{j=1}^{K^*} \sum_{i \in \mathcal{V}_j} \exp(\alpha_{0i}) \left[\|\alpha_{1i} - \alpha_{1j}^*\| + \|\beta_i - \beta_j^*\| + |\sigma_i^2 - \sigma_j^{*2}| \right], \end{aligned} \quad (36.10)$$

where we denote Above, if a Voronoi cell $\mathcal{V}_j$ is empty, then we let the corresponding summation term be zero. Additionally, it can be checked that $\mathcal{L}(G, G^*) = 0$ if and only if $G \equiv G^*$. Thus, when $\mathcal{L}(G, G^*)$ is sufficiently small, the differences $\alpha_{1i} - \alpha_{1j}^*$, $\beta_i - \beta_j^*$, and $\sigma_i^2 - \sigma_j^{*2}$ are also small. This property indicates that $\mathcal{L}(G, G^*)$ is an appropriate loss function for measuring parameter discrepancies. However, since the loss $\mathcal{L}(G, G^*)$ is not symmetric, it is not a proper metric. Lastly, computing the Voronoi loss function $\mathcal{L}(G, G^*)$ is efficient as its computational complexity is at the order of $\mathcal{O}(K^{*2})$. Note that the key benefits of Voronoi-based loss functions are expressed more clearly in the overspecified setting where the rates for estimating overspecified parameters (parameters of G^* fitted by at least two parameters) are slower than those for estimating exactly-specified parameters (parameters of G^* fitted by one parameter) [1238]. In particular, Nguyen et al. [1238] have shown that Voronoi-based loss functions are capable of characterizing those rates, while the generalized Wasserstein used in Ho et al. [756] are incapable of capturing the rate difference.

Given the Voronoi-based loss in equation (36.10), we are ready to present results on the convergence rate of parameter estimation in the following theorem.

Theorem 36.2. *Suppose that Assumptions 36.1–36.5 hold true. When the number of ground-truth experts K^* is known, we find that*

$$d_H(g_G, g_{G^*}) \gtrsim \mathcal{L}(G, G^*), \quad (36.11)$$

for all mixing measures of the form $G = \sum_{j=1}^{K^*} \exp(\alpha_{0j}) \delta_{(\alpha_{1j}, \beta_j, \sigma_j^2)}$. As a consequence, the Hellinger lower bound (36.11) and the result of Theorem 36.1 together imply that for any sequence $M_n \rightarrow \infty$, we have

$$\lim_{n \rightarrow +\infty} \Pi \left(\left\{ G \in \mathcal{G}_K : \mathcal{L}(G, G^*) \gtrsim M_n \sqrt{\log(n)/n} \right\} \mid (X_i, Y_i)_{i=1}^n \right) = 0. \quad (36.12)$$

in $g_{G^*}^n$ -probability.

A few remarks regarding the results of Theorem 36.2 are in order.

(i) *Convergence rate of parameter estimation:* From equation (36.12) and the formulation of the Voronoi-based loss $\mathcal{L}(G, G^*)$ in equation (36.10), it follows that the convergence rates of estimations of parameters $\exp(\alpha_{0j}^*)$, α_{1j}^* , β_j^* , and σ_j^{*2} admit the same order of $\mathcal{O}_P(\sqrt{\log(n)/n})$, which is parametric on the sample size n .

(ii) *Convergence rate of expert estimation:* Assume that the expert function $\zeta(x, \beta)$ is Lipschitz continuous with respect to its parameter β , which yields

$$|\zeta(x, \beta_i) - \zeta(x, \beta_j^*)| \lesssim \|\beta_i - \beta_j^*\|,$$

for all $i \in \mathcal{V}_j$ and $j \in [K^*]$. It should be noted that the above Lipschitz continuity condition is satisfied by several expert choices in practice, including two-layer feed-forward networks used in the Transformer architecture in large language models [1669]. Then, from the rate for estimating parameter β_j^* in part (i), we deduce that the estimators of experts $\zeta(x, \beta_j^*)$ also share the convergence rate of the parametric order $\mathcal{O}_P(\sqrt{\log(n)/n})$, for $j \in \{1, 2, \dots, K^*\}$. As a consequence, to estimate an expert $\zeta(x, \beta_j^*)$ with a given error $\epsilon > 0$, we need a polynomial number of data points $\mathcal{O}(\epsilon^{-2})$.

Overspecified setting. Next, we extend the above analysis to the scenario in which the number of ground-truth experts K^* becomes unknown. A popular approach to this case is to over-specify the ground-truth model in equation (36.4) by a softmax-gated Gaussian mixture of K experts, where $K > K^*$. Then, according to the pigeonhole principle, when this fitted model becomes sufficiently close to the ground-truth model, at least one of the ground-truth components $(\alpha_{11}^*, \beta_1^*, \sigma_1^{*2}), \dots, (\alpha_{1K^*}^*, \beta_{K^*}^*, \sigma_{K^*}^{*2})$ will be approximated by more than one fitted component. Thus, there exists a Voronoi cell $\mathcal{V}_j$, where $j \in \{1, 2, \dots, K^*\}$, having more than one element. As a result, the Voronoi-based loss in equation (36.10) cannot be applied to the overspecified setting, indicating that

the convergence rates of parameter and expert estimations will be affected. In particular, Nguyen et al. [1238] have shown that these rates depend on the cardinality of Voronoi cells, thereby becoming substantially slower than the parametric rate of order $\mathcal{O}_P(\sqrt{\log(n)/n})$ achieved in the exactly-specified setting. However, due to the technical complexity of this analysis, which is beyond the scope of this chapter, we would rather refer the reader to Nguyen et al. [1238] rather than present it in further detail here.

36.5 Model selection

An important open question in Bayesian MoE models is how to choose the number of experts K before training. As a rough rule of thumb, setting K large ensures either $K \geq K^*$ (if the true data-generating process is MoE), so the true density is included, or, under misspecification, makes the model rich enough to approximate the unknown distribution.

However, selecting a very large K has drawbacks. First, the parameter space grows linearly with K , making training computationally expensive, especially with high-dimensional expert or gating functions. Second, over-specification harms parameter estimation accuracy. Third, in applications where inference and interpretability matter, fewer parameters aid both clarity and uncertainty quantification. Thus, it is desirable to have a *model selection* procedure to choose the “best” K , where “best” depends on the application.

A pragmatic approach is to define “best” as what works in practice. In deep learning, MoE layers scale Transformer models [392, 603, 843], but computational limits make selecting the optimal K critical. Ludziejewski et al. [1089] proposed a scaling law relating final loss, training data, model size, and K , validated through extensive experiments. This enables selecting K given other hyperparameters. Scaling laws for MoE-based LLMs are further explored in Abnar et al. [5], Tian et al. [1605].

Alternatively, under the assumption that the true model is MoE with K^* experts, one may attempt to recover K^* via statistical procedures. We present two such strategies next, discussing their advantages and limitations.

36.5.1 K as a learnable parameter

In the Bayesian framework we work with, it is natural to treat any aspect of the data-generating process we want to learn about as a parameter to be estimated for the data. In particular, quantities that are part of the likelihood model can be learned from data by including them in the Bayesian model to be trained, and in particular by assigning a prior to them and computing the resulting posterior distribution.

This simple principle can be readily applied to the number of experts K , as follows. Consider the usual likelihood model

$$Y_i | X_i, K, G \stackrel{\text{ind}}{\sim} f_G(\cdot | X_i),$$

recalling how, for notational convenience, we collect the MoE parameters into the measure $G = \sum_{j=1}^K \exp(\alpha_{0j}) \delta_{(\alpha_{1j}, \beta_j, \sigma_j^2)}$ with K support points, one for each expert. The key in the above expression is to explicitly recognize the functional dependence of the distribution of Y_i on K , by treating the latter as one more parameter in the model (as the ones collected by G). The next step is to assign priors, starting with G :

$$G | K \sim \Pi_K$$

where Π_K is a probability distribution on the set of positive measures with K support points on the space of expert parameters. Once again, while this formulation may sound abstract, it simply means that Π_K is a prior for the parameters of an MoE model with K experts. The crucial point is that we are now explicitly acknowledging the number of experts in the MoE model construction, while we left this information implicit when we were previously fixing K to an arbitrary but large number. Finally, because we want to learn K , we model it by assigning a prior ϕ (supported on $\mathbb{N}$) to it as well: $K \sim \phi$.

Having set up the model, we compute the posterior in the usual way as

$$\Pi(G, K | Y_{1:n}, X_{1:n}) \propto \phi(\{K\}) \times \Pi_K(G) \times \prod_{i=1}^n f_G(Y_i | X_i), \quad (36.13)$$

where the product of the first two terms on the right-hand side of the previous expression is simply the factorization of the joint prior for G and K implied by the hierarchical model. An important fact to notice is that the dimension of the support of the joint posterior distribution (36.13) varies with K : the larger K , the more parameters the model includes. This implies that, whatever learning algorithm one employs (whether MCMC or variational inference), this variable-dimensional nature of the parameter space should be taken care of appropriately. We do

not dwell on this aspect any further and refer the reader to the rich literature on the topic of a random number of mixture components (mostly developed with traditional mixtures in mind).

If we want to make inference on K , we can compute the marginal posterior distribution of K by marginalizing (36.13):

$$\Pi(K \mid Y_{1:n}, X_{1:n}) = \int \Pi(dG, K \mid Y_{1:n}, X_{1:n}).$$

Notice that, in practice, the marginal posterior is usually available (approximately) as a byproduct of the training algorithm we employ to fit the model; for instance, if we use an MCMC sampler where K is resampled at each iteration, we can simply take the empirical distribution of the marginal Markov Chain as an approximation to the marginal posterior of K . Be it as it may, once we have access to $\Pi(K \mid Y_{1:n}, X_{1:n})$, we can use it to make inference on K . For example, we can obtain a point estimate by computing the mode of the marginal posterior, and we can get uncertainty quantification by selecting as a credible set of values those that achieve the highest posterior probabilities which sum up to at least $1 - \kappa$ (where $\kappa \in (0, 1)$ is some small tolerance level).

An interesting question to ask is whether treating K as a parameter in the Bayesian model leads to good empirical performance, at least when enough data is collected. A way to study this aspect, as we did in the previous section, is to assume that the data is generated from a proper MoE model (36.1) with a fixed but unknown number of experts K^* , and to study whether the posterior distribution concentrates on the correct value: that is, whether

$$\Pi(K^* \mid X_{1:n}, Y_{1:n}) \xrightarrow{n \rightarrow \infty} 1$$

in some appropriate probabilistic sense.

To facilitate the discussion, let's now fix some notation. Recall that Θ denotes the space of parameters relative to a single expert (i.e., the product of the space of gating parameters (α_0, α_1) and the space of expert parameters (β, σ^2)), and assume that the parameters for each different expert all belong to Θ . Therefore, when we look at an MoE model with k experts, the overall parameter space coincides with $\Theta^k := \Theta \times \cdots \times \Theta$ (k times). In our case, however, we are looking to place a prior that places positive probability on MoE models with any $K \in \mathbb{N}$, so the parameter space of interest is $\Theta_\infty := \bigcup_{k \in \mathbb{N}} \Theta^k$. Accordingly, the prior Π will be a probability measure on Θ_∞ specified hierarchically as follows: $\theta \sim \Pi$ if and only if $\theta \mid K \sim \Pi_K$, $K \sim \phi$, where Π_K is a probability measure on Θ^K . That is, to draw a random parameter θ from the prior Π , one first draws a random number of experts $K \sim \phi$ and then, conditional on this draw, one samples a K -expert MoE parameter from Π_K .

With this setting in mind, we are ready to state the next result about the asymptotic consistency of the posterior distribution on the number of experts.

Theorem 36.3. *Assume that Θ is compact, with the space of variances bounded away from 0. Moreover, assume the following:*

1. *The MoE model is identifiable;*
2. *Let Π_Θ be a continuous probability measure on Θ such that $\Pi_K = \Pi_\Theta^K$ (i.e., conditional on the number of experts K , the prior is of product form across experts);*

Then there exists a set $\Theta_\star \subseteq \Theta_\infty$ such that $\Pi(\Theta_\star) = 1$ and, for all $\theta_\star \in \Theta_\star$,

$$\lim_{n \rightarrow \infty} \Pi(K(\theta_\star) \mid X_{1:n}, Y_{1:n}) = 1$$

almost surely under $g_{G(\theta_\star)}^\infty$, where, for any $\theta \in \Theta_\infty$, $K(\theta)$ and $G(\theta)$ denote, respectively, the number of experts and the mixing measure associated to θ .

The proof of the last result, which we omit here but can be found in Bariletto et al. [97], is based on a fundamental result in Bayesian posterior asymptotics, namely Doob's consistency theorem. Loosely speaking, Doob's theorem says that, as long as a parameter of interest can be exactly recovered with infinite data, the posterior distribution is consistent for a set of such parameters that has prior mass 1. The result says, then, that this is the case for the number of experts in the MoE model (under the stated assumptions). Therefore, as long as the true generating MoE model lies in such a set of prior probability 1,⁵ posterior inference on the number of experts will lead to the correct answer as the sample size grows to infinity.

⁵The fact that such a set has prior probability 1 means that, when looked from the perspective of the prior distribution, this set is large. However, this interpretation should be taken with some caution, as sets of prior mass 1 may anyway exclude large portions of the parameter space, especially in infinite-dimensional settings.

36.5.2 Selection via the ELBO

Another strategy to select the number of experts K relies on the output of the training process when variational inference is used. In particular, at the end of the training loop, on top of an optimized parameter vector one obtains (an estimate of) the evidence lower-bound of the trained model. As the name itself reminds us, the evidence lower-bound can be seen as a proxy of the evidence (or marginal likelihood) associated to the model, that is

$$M_n(\Pi) := \int \prod_{i=1}^n f_G(Y_i | X_i) \Pi(dG).$$

Moreover, recall that $M_n(\Pi)$ is often used in Bayesian theory as a key quantity to compare prior models Π and select the best one based on how high the corresponding $M_n(\Pi)$. This makes intuitive sense because $M_n(\Pi)$ measures the likelihood computed at the training sample, integrated against the prior—so the higher is this quantity, the better is the fit of Π .

Piecing together the previous observations, a simple algorithm arises as a natural criterion for selecting the number of experts K :

1. Choose a set $\mathcal{K} \subseteq \mathbb{N}$ of candidate values of K , e.g., those that are more plausible given the application domain of the learning task;
2. For each $K \in \mathcal{K}$, fit an MoE model with K experts and record the value of the evidence lower-bound at the end of training, $\text{ELBO}(K)$;
3. Select the best K as the one maximizing the evidence lower-bound:

$$K_{\text{best}} = \arg \max_{K \in \mathcal{K}} \text{ELBO}(K).$$

Just like when we proposed placing a prior on K , we can ask ourselves if the above procedure produces a consistent estimate of K^* , under the assumption that the data-generating process is an MoE. That is, we can ask if, as the sample size n grows to infinity, the estimator K_{best} converges to K^* in some appropriate probabilistic sense. A theoretical analysis of this procedure has been carried out in the context of simpler models, for instance mixtures [1703, 1862], but is still lacking for models with more complicated likelihoods such as MoE ones. Nevertheless, some preliminary empirical evidence is presented in Bariletto et al. [97].

36.6 Extensions and future directions

There are two potential future directions that can be extended from the above studies of Bayesian MoE models.

1. *Bayesian sparse MoE.* In large-scale deep learning applications such as large language models, MoE is often utilized to scale up the model capacity (the number of model parameters) while keeping the computational overhead nearly unchanged. In particular, Shazeer et al. [1508] proposed a sparse variant of MoE where only a few experts were activated per input. For that purpose, a sparse gating function or router was employed to evaluate the relevance of experts to an input, and then selected a subset of experts which had the highest similarity scores. Despite its huge success in practice, the sparse gating function induces several technical challenges in studying the asymptotic properties and the model selection problem of sparse MoE due to its discontinuous nature. From the perspective of frequentists, Nguyen et al. [1240, 1243] addressed these challenges by partitioning the input space into multiple subregions, each of which corresponded to a subset of experts activated per input in that region. Next, they characterized the asymptotic properties of sparse MoE in each region and then unified these results to capture the properties of sparse MoE in the whole region. On the other hand, the technical challenges caused by the discontinuity of the sparse gating function have remained unsolved from the Bayesian perspective.

2. *Bayesian nonparametric MoE.* At an abstract level, Bayesian mixtures of experts (MoEs) can be viewed as a special case of dependent nonparametric mixture models [1103], where the mixing measures depend on covariates. The key distinction lies in their goals: dependent nonparametric mixtures (e.g., the dependent Dirichlet process [1356]) are primarily designed for density estimation, whereas Bayesian MoEs are tailored to estimating expert models (e.g., regression functions). The two approaches also differ in how they construct mixture weights. Dependent nonparametric mixtures typically define weights through dependent stick-breaking priors, while Bayesian MoEs place a prior directly on the gating function (e.g., a softmax) that governs the mixture weights. In this sense, Bayesian MoEs are conceptually similar to the construction in Dunson et al. [464], where a kernel is used to define a data-dependent mixing measure. The main difference, however, is that in Dunson et al. [464] the kernel measures

similarity between data points, whereas in Bayesian MoEs the gating function computes similarity between a data point and model parameters.

A practical consideration for Bayesian MoEs is the choice of prior $\Pi(G, K)$. As discussed in Section 36.5.1, we factorize it as $\Pi(K)\Pi_K(G)$. Alternatively, one may specify a joint prior directly over (G, K) . For example, a repulsive prior on G can be employed [1319] to discourage redundant experts. More generally, $\Pi(G, K)$ can be modelled via a determinantal point process [944, 1102], where experts are treated as stochastic points. The corresponding inference procedures under such specifications also merit further investigation, with reversible jump methods [662] being a conventional approach. For the gating function and expert functions, we can go beyond parametric construction by using Gaussian process [1381, 1631].

Part VIII

Scalability

Chapter 37

Scalable BDL by single forward pass inference

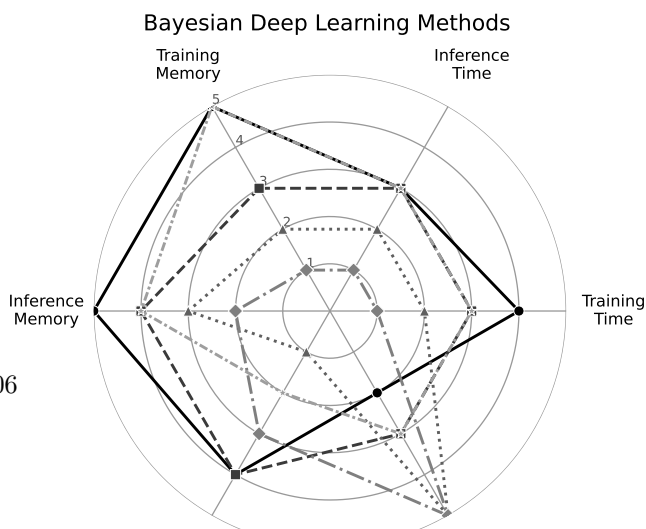
In the previous sections, we surveyed a broad range of technical methods that bridge the Bayesian framework with deep learning. This section takes a different angle, perhaps from a more practical standpoint, to investigate the following question: *How can we devise scalable methods for Bayesian Deep Learning?* We begin by motivating why thinking about scalability is essential for tackling problems of relevance with BDL. We then turn our attention to a specific class of scalable BDL methods that enable inference with a single forward pass through a deep network. These methods can be naturally understood as the composition of a *backbone* network, intended for representation learning, and a *predictive head*, which performs kernel approximation or density estimation for scalable Bayesian inference. Throughout this chapter, we discuss the challenges associated with each component and examine the distinct methods in the literature to address them.

37.1 Why do we need scalability in Bayesian Deep Learning?

So far, this book has presented a plethora of BDL methods, encompassing diverse approaches to enable Bayesian inference with deep learning models. These methods adhere to the formal language of Bayesianism, which offers a principled and self-consistent framework for probabilistic machine learning. They also provide important capabilities that non-Bayesian approaches typically lack, such as uncertainty estimation. However, Bayesian inference (whether exact or approximate) requires integrating over high-dimensional parameter spaces, which is computationally demanding. As a result, many BDL methods struggle to scale to the large models and datasets common in modern applications. Striking the right balance between computational efficiency, stability, and approximation quality remains a central challenge and an active area of research.

Over the past decade, deep learning research has been shaped by two defining characteristics. First, the prevalence of *large-scale datasets* from the Big Data era, composed of millions (or perhaps billions or even trillions) of data points. For instance, ImageNet [1441]—once a standard computer vision benchmark—contains more than 14 million data points; LAION-5B [1475], a recent multimodal dataset, contains 5.85 billion image-text pairs. Second, modern models are *highly overparameterised*, often containing millions to billions of parameters. Even relatively modest architectures, such as ResNet-18 [721], have 11.8 million parameters, while state-of-the-art foundation models, including LLMs/VLMs, can reach hundreds of billions or even trillions. Consequently, methods that fail to scale in memory or compute time may not be effective in practice.

We now turn to the memory and time complexities of the key methods discussed in this book. For clarity, our comparison focuses on their “vanilla” versions, acknowledging that subsequent refinements may improve scalability. Table 37.1 summarises time and memory complexities during training and inference stages, while Figure 37.1 extends the comparison along two additional dimensions: *ease of use*, which refers to how straightforward a method is for implementation, training, debugging, or tuning; and *approximation quality*, which refers to how well it captures posterior distri-



Method	Training Time	Inference Time	Training Memory	Inference Memory
Standard NN	$\mathcal{O}(N \cdot P)$	$\mathcal{O}(P)$	$\mathcal{O}(P)$	$\mathcal{O}(P)$
MC Dropout	$\mathcal{O}(N \cdot P)$	$\mathcal{O}(M \cdot P)$	$\mathcal{O}(P)$	$\mathcal{O}(P)$
Deep Ensembles	$\mathcal{O}(M \cdot N \cdot P)$	$\mathcal{O}(M \cdot P)$	$\mathcal{O}(M \cdot P)$	$\mathcal{O}(M \cdot P)$
Variational Inference	$\mathcal{O}(K \cdot N \cdot P)$	$\mathcal{O}(K \cdot P)$	$\mathcal{O}(P)$	$\mathcal{O}(P)$
MCMC	$\mathcal{O}(T \cdot N \cdot P)$	$\mathcal{O}(S \cdot P)$	$\mathcal{O}(S \cdot P)$	$\mathcal{O}(S \cdot P)$
Gaussian Processes	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$

Table 37.1: Complexity comparison of BDL methods, including a standard neural network baseline. Here, N and P denote the number of training points and model parameters. M , K , T and S correspond to the number of ensemble/dropout samples, VI samples, MCMC steps, and MCMC posterior samples, respectively. Typically, $T \gg S > K \gtrsim M$.

butions. Although this plot is inevitably subjective, it serves as a useful didactic illustration.

Methods originally designed with deep learning practice in mind (such as Monte Carlo (MC) Dropout and Deep Ensembles) present time and memory complexities that more closely resemble those of standard neural networks, suggesting better scalability. They are also easily integrable to deep learning pipelines, demonstrating good practicality. Yet, they present drawbacks. MC Dropout, despite being theoretically justified as a variational approximation (Chapter 14), assumes a restrictive variational family (a degenerate spike-and-slab distribution), which hinders its approximation quality and sometimes its predictive performance. Deep Ensembles (Chapter 19), on the other hand, often achieve strong empirical performance and reasonable uncertainty estimates, but incur significantly higher computational costs than a single deterministic network, as they require training and maintaining M independent model copies. Furthermore, their theoretical foundations in the Bayesian framework are still not completely understood (and a topic of disagreement among Bayesians), which leaves open the question of why they work and when their uncertainty estimates should be trusted.

Variational inference (VI) methods typically offer a compromise between computational efficiency and approximation quality. Compared to MC Dropout, they allow for richer variational families; yet, in practice, the distributions used remain highly restrictive to preserve tractability in the learning objective (e.g., mean-field VI). Unlike Deep Ensembles, they do not require training multiple copies of a model and can be integrated relatively easily into standard deep learning pipelines, which has made modern VI approaches both promising and widely adopted in BDL research. Nevertheless, training stability remains a significant challenge, often demanding careful hyperparameter tuning, especially for deeper or more complex architectures. Perhaps as a consequence, it is also unclear how to reconcile these methods with the emerging paradigm of Foundation Models.

Finally, we consider Bayesian methods that offer the most principled inference guarantees. Gaussian Processes (GPs) provide exact inference for a chosen kernel under certain likelihood models, and their closed-form analytical solutions are appealing to practitioners in many settings. However, their computational cost grows cubically with the dataset size, making direct application to modern deep learning problems infeasible. Sampling-based methods such as Markov chain Monte Carlo (MCMC) are asymptotically exact (that is, they theoretically converge to the true posterior in the limit) and thus provide the most faithful approximations available in high-dimensional problems. Yet, they are notoriously slow and memory intensive, requiring a large number of MCMC samples T for convergence and posterior samples S for inference, and are often tricky to implement and tune effectively.

In summary, a key takeaway is that *there is no silver bullet for Bayesian Inference in Deep Learning*. While the baseline methods presented here have their strengths, none of them really demonstrate the scalability and simplicity of standard deep networks. They exhibit a clear tension between algorithmic complexity (both in

terms of memory and time) and approximation quality or, more broadly, practicality. This is why developing scalable methods is crucial: to reconcile these trade-offs and make Bayesian approaches viable within modern deep learning pipelines and real-world applications. In this chapter, as well as the subsequent ones, we will explore various strategies to advance scalability in Bayesian Deep Learning.

37.2 Bayesian Inference by a single forward pass

Let us once again turn our attention to Table 37.1, particularly on parametric methods (we will explore GPs later). The first thing to notice is that the costs of all of them, including standard NNs, grows linearly with the number of model parameters. This cost looks non-negotiable, as we require to at least compute the model predictions by running over them. It is also not negligible, as the current model are already very large and the trend in the field is to gradually increase their size further. In this context, what can we do to improve scalability?

The natural answer is to focus on the other variables that affect their complexity. A common pattern among them is the presence of a multiplicative factor, which is related to the process of marginalisation. For instance, let us look at the process of predictive inference, which involves approximating the following expectation:

$$p(y \mid x, \mathcal{D}) = \int p(y \mid x, \theta) p(\theta \mid \mathcal{D}) d\theta = \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D})} [p(y \mid x, \theta)]. \quad (37.1)$$

In most cases of BDL, the expectation above is approximated via Monte-Carlo sampling from a posterior approximation $q(\theta) \approx p(\theta \mid \mathcal{D})$. Each posterior sample θ is a *model hypothesis*: a different “version” that the neural network may assume. For instance, these are the M dropout masks in MC dropout, or the M models in Deep Ensembles, or the S posterior samples in MCMC, and so on. This means that most methods will require running multiples “versions” of an NN for a single inference, sometimes requiring storing a copy of each of them in memory. This explains why these methods are much more costly than standard NNs. Even a low number of samples (for instance, we often use $M = 5 \sim 10$) incurs high costs as the deep models are naturally very large.

We highlight two different strategies to circumvent these challenges. The first one is to assume a particular structure in the posterior distribution that allows computing it and approximating Equation 37.1 with lower cost, perhaps in a closed-form solution. One way of doing this is via a *Laplace approximation* (Chapter 7), which assumes a Gaussian posterior centred around the MAP estimate. With further approximations (diagonal approximation of Hessians, model linearisation), it is possible to achieve time and memory complexities equivalent to a standard NN. Naturally, these approximations affect the quality of inference. We explore scalable Laplace approximations in Chapter 40.

Another alternative, more relevant for this chapter, is to *accept being Bayesian only in a subspace of the model parameters*. Let $\theta = [\theta_b, \theta_h]$, where b and h refers to a model’s *backbone* and *head*, respectively, where $P = \dim(\theta) = P_b + P_h$. For a Bayesian method with complexity $\mathcal{O}(M \cdot P)$, if we opt to be Bayesian only in the head parameters, we can leverage the output of the backbone as representations (via a single forward pass) and compute the marginalisation of Equation 37.1 in the Bayesian head. In this case, the complexity will be $\mathcal{O}(P_b + M \cdot P_h)$. For the case where $P_b \gg P_h$ and for small number of posterior samples M , $\mathcal{O}(P_b + M \cdot P_h) \approx \mathcal{O}(P_b) \approx \mathcal{O}(P)$. As an extreme case, we may consider a non-parametric head, such as a GP (over a small set of points), attaining complexity of $\mathcal{O}(P + N^3)$, again with $P \gg N$ so that it approximates $\mathcal{O}(P)$. This idea is explored both in this chapter as well as in Last-Layer Inference methods (Chapter 39).

37.2.1 Formalism

We now formalise the idea of BDL with a single forward pass inference. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the input space and $\mathcal{Y} \subseteq \mathbb{R}^q$ the output space. We are given training data with N data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, assumed to be generated from an unknown stochastic process $y \sim p(y \mid f^*(\mathbf{x}))$, $f^* : \mathcal{X} \rightarrow \mathcal{Y}$, where f^* is the true (unknown) function underlying the observations. The learning objective is to approximate the predictive distribution $p(y_* \mid \mathbf{x}_*, \mathcal{D})$. A BDL method with single forward pass inference can be described by two components:

- A *backbone* network, defined as a parametric map $g_{\theta_b} : \mathcal{X} \rightarrow \mathbb{R}^p$, where θ_b represents the trainable parameters of this backbone (e.g., a deep neural network). It maps the input space (usually high-dimensional) to a low-dimensional representation space. The goal is to leverage the inductive biases and expressiveness of the deep network to learn high-level representations for probabilistic modelling.
- A *predictive head*, defined as a functional mapping $\mathcal{U}_{\theta_h} : \mathbb{R}^p \rightarrow \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes the set of probability distributions over $\mathcal{Y}$. As we mentioned previously, this mapping may be a parameterised function

(by θ_h), or assume a non-parametric form (as in the case of a GP head). This head governs predictive uncertainty, via a kernel function or a density estimator.

Two seminal works in the BDL world to apply these ideas are Deep Kernel Learning (DKL) [1769], discussed in more detail in Chapter 24, and Deterministic Uncertainty Quantification (DUQ) [1653]. In DKL, the backbone network extract features over which an inducing point GP is defined. We can think about it as defining a new kernel as:

$$k_{\theta_b}(\mathbf{x}_i, \mathbf{x}_j) \rightarrow \bar{k}(g_{\theta_b}(\mathbf{x}_i), g_{\theta_b}(\mathbf{x}_j)), \quad (37.2)$$

where $\bar{k}$ is a base kernel, such as the RBF kernel. DUQ is a “non-GP” counterpart to DKL: instead of modelling a GP on top of the deep kernel, it performs predictions and estimate uncertainties directly in the learned feature space via the kernelized distance (i.e., a RBF network).

37.3 The challenge of a good backbone

As described previously, the goal of the backbone is to leverage the expressive power of deep networks to learn good features that improve predictions. Nonetheless, learning this feature space is not a simple task, and naively following standard modelling assumptions sets up an underconstrained optimisation that potentially leads to pathological, trivial solutions, where the learned representations are degenerated. For instance, the backbone may learn to map dissimilar points to the same representation, a phenomenon called “feature collapse” [1653, 1654]. This is often the case for out-of-distribution data, where these learning objectives do not impose restrictions to points that are away of the training distribution.

Representation degeneracy poses a particular challenge for BDL methods because these methods rely on the kernel to encode prior assumptions about the function space. In practice, most kernels used in this setting are *geometry-aware*: they measure similarity between inputs through pairwise distances in the learned feature space, $d(\mathbf{x}, \mathbf{x}') = \|g_{\theta_b}(\mathbf{x}) - g_{\theta_b}(\mathbf{x}')\|$. If the feature space fails to represent meaningful geometry, the kernel matrix also degenerates. As a result, the probabilistic model can no longer provide reliable predictions and uncertainty estimates.

A concrete example where this pathology arises is in Deep Kernel Learning. DKL maximises the log marginal likelihood of a GP with prior $\mathcal{GP}(m, K)$, where K is the kernel matrix and takes the form $K_{i,j} = k_{\theta_b}(x_i, x_j)$, with k_{θ_b} as described in Equation 37.2. Mathematically, this objective is given by:

$$\begin{aligned} \log p(\mathbf{y}) &= \log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, K + \sigma_n^2 I_N) \\ &= - \underbrace{\frac{1}{2} \mathbf{y}^\top (K + \sigma_n^2 I_N)^{-1} \mathbf{y}}_{\text{data fit term}} - \underbrace{\frac{1}{2} \log |K + \sigma_n^2 I_N|}_{\text{complexity term}} - \underbrace{\frac{N}{2} \log(2\pi)}_{\text{constant}}, \end{aligned} \quad (37.3)$$

where σ_n is a learnable hyperparameter for the observation noise. In addition to the constant term, we identify two main components of this objective: the *data fit* term, a function of the observables $\mathbf{y}$ that optimises the kernel parameters to better explain the data; and a *complexity* term, which encourages simpler models.

At convergence, one can show that the data-fit term approaches $-N/2$ (this follows from the stationarity condition of the log marginal likelihood with respect to a global scale on $K + \sigma_n^2 I_N$: at the optimal scale, $\mathbf{y}^\top (K + \sigma_n^2 I_N)^{-1} \mathbf{y} = N$ exactly, so the data-fit term evaluates to $-N/2$). More interestingly, the complexity term encourages minimising the determinant $|K + \sigma_n^2 I_N|$. This determinant can be driven arbitrarily small by learning a kernel that maximally correlates the input representations without compromising the data fit—something feasible when the kernel has sufficient capacity, as in the case of deep backbone networks. A trivial solution arises when the kernel induces co-linear representations, driving the determinant toward zero and effectively collapsing the features. In this sense, the objective *encourages* feature collapse when using highly parameterised kernels.

We empirically illustrate this phenomenon in Figure 37.2, using a 2D classification task where the two classes are generated from Gaussian distributions. Figure 37.2a shows a grid of input points, coloured according to their log-probability under the data-generating distribution. In contrast, Figure 37.2b displays the learned representations obtained from an underconstrained model. Here, the points collapse into nearly co-linear representations, regardless of their original positions or likelihoods. This pathology leads to undesirable consequences, such as overfitting [1253] and poor uncertainty estimation [1654], even within a Bayesian framework.

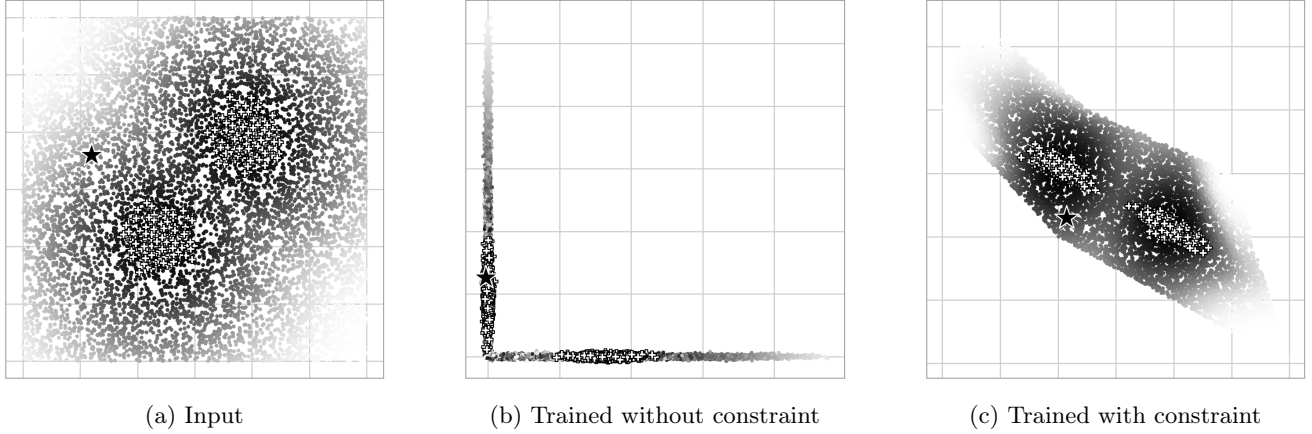


Figure 37.2: A 2D classification task where the classes are two Gaussians (marked with white “+” symbols), and a grid of unrelated points, coloured according to their log-probability under the data generating distribution. A specific point is marked with a star. In (b), the features as computed by an unconstrained model. In (c), the features computed by a model with residual connections and spectral normalisation. The unconstrained model distorts the space, collapsing inputs onto a line regardless of their dissimilarities. In contrast, the constrained mapping preserves relative distances among points. Adapted from van Amersfoort et al. [1654].

37.4 The Antidote for the degeneracy: bi-Lipschitzness

How can we prevent representation degeneracy? One natural strategy is to return to the full Bayesian setting [1253] and place priors on all kernel parameters to prevent pathological solutions. Obviously, this approach defeats our purpose of making BDL more scalable, as discussed in Section 37.1. Instead, we can take an optimisation perspective and view feature collapse as the result of an *underconstrained problem*, and address it by imposing regularisation assumptions on the learned feature space to prevent such pathological solutions.

As we previously discussed, representation degeneracy is a major concern since kernel-based predictive heads usually rely on the geometry of the feature space, and, in particular, the induced distance between points $d(\mathbf{x}, \mathbf{x}') = \|g_{\theta_b}(\mathbf{x}) - g_{\theta_b}(\mathbf{x}')\|$. Therefore, preventing degeneracy encompasses learning a mapping that is *distance-preserving*, so that the distance in the feature space $d(\mathbf{x}, \mathbf{x}')$ has a meaningful correspondence to the distance $\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}$ in the data manifold [1061]. Here, $\|\cdot\|_{\mathcal{X}}$ denotes a semantically meaningful metric on the input data manifold, in contrast to the Euclidean norm in the raw observation space, which is a poor proxy for semantic similarity in high-dimensional inputs such as images.

We can decompose the distance-preserving property into two conditions. First, we require that different points $\mathbf{x}, \mathbf{x}'$ also present different representations: when the input changes, the features also change. This condition is called *sensitivity*, and one can express it mathematically as follows:

$$L_1 \leq \frac{\|g_{\theta_b}(\mathbf{x}) - g_{\theta_b}(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}' \quad (37.4)$$

where $0 < L_1 < 1$. Equation 37.4 is often referred as the lower Lipschitz condition, and L_1 is the lower Lipschitz constant. Intuitively, if we fix two distinct points $\mathbf{x}, \mathbf{x}'$, then this constraint prevents the numerator (the distance in feature space) to become too small.

The second condition for a distance-preserving mapping is *smoothness*: small changes in the input should not lead to arbitrarily large changes in the representations. Smoothness complements sensitivity by imposing an upper Lipschitz condition:

$$L_2 \geq \frac{\|g_{\theta_b}(\mathbf{x}) - g_{\theta_b}(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}', \quad (37.5)$$

where $L_2 > 1$ is the upper Lipschitz constant. Note that, by definition, $L_1 < L_2$. The composition of both conditions is called the bi-Lipschitz condition, and often expressed as:

$$L_1 \|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}} \leq \|g_{\theta_b}(\mathbf{x}) - g_{\theta_b}(\mathbf{x}')\| \leq L_2 \|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}'. \quad (37.6)$$

Next, we highlight different strategies to enforce bi-Lipschitzness during deep networks training.

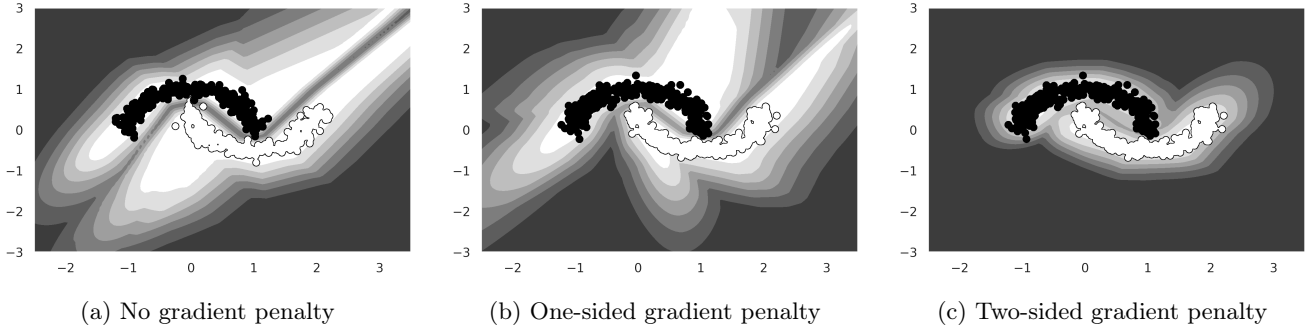


Figure 37.3: Uncertainty quantification on the two moons dataset using an RBF network. Lighter shading indicates high certainty, while darker shading indicates uncertainty. Ideally, the model should be confident only near the training data. In (a), the model lacks any regularisation and remains overconfident far from the data. In (b) and (c), adding one-sided and two-sided penalties improves uncertainty estimates, with the bi-Lipschitz penalty in (c) giving the most reliable results. Adapted from Van Amersfoort et al. [1653].

37.4.1 Bi-Lipschitzness via Two-Sided Gradient Penalty

From the fundamental theorem of calculus, one can show that, for $f : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}^k$ continuously differentiable on a convex set Ω :

$$\|f(a) - f(b)\| \leq \sup_{z \in [a, b]} \|J_f(z)\| \cdot \|a - b\|, \quad (37.7)$$

where J_f is the Jacobian. The Jacobian norm $\|J_f(z)\|$ provides a local bound on how much f can expand distances near z . Indeed, if $\|J_f(z)\| \leq L_2$ for all $z \in \Omega$, then we recover the upper Lipschitz condition. On the other hand, if the smallest singular value of the Jacobian is uniformly bounded below as $\sigma_{\min}(J_f(z)) \geq L_1 > 0$, we also recover the lower Lipschitz condition, since for any vector v , $\|J_f(z)v\| \geq \sigma_{\min}(J_f(z)) \|v\|$. Together, these inequalities establish that the global Lipschitz constants of f can be controlled directly by regularising the Jacobian norm. A simple way to achieve this is to add a penalty term that explicitly constrains both the largest and the smallest singular values of the Jacobian:

$$\lambda (\max(0, \|J_f(z)\| - L_2))^2 + \mu (\max(0, L_1 - \sigma_{\min}(J_f(z))))^2, \quad (37.8)$$

where λ and μ are hyperparameters. This objective enforces that the singular values remain in the interval $[L_1, L_2]$, thereby directly encouraging the bi-Lipschitz property. In practice, however, previous works [686, 1653] adopt a simpler penalty of the form

$$\lambda (\|J_f(z)\|^2 - 1)^2. \quad (37.9)$$

While this penalty formally only controls the largest singular value, in practice it often has the effect of improving the overall conditioning of the Jacobian, discouraging collapse in the smaller singular values as well.

Figure 37.3 presents an example of the effectiveness of enforcing bi-Lipschitzness by gradient penalty. In this example from Van Amersfoort et al. [1653], a RBF network is trained on the two moons dataset, a 2D classification task, and we evaluate how well the model performs uncertainty quantification for points that are out-of-distribution. Ideally, the model should be confident only near the training points. In Figure 37.3a, without any penalty enforcing smoothness or sensitivity, the network remains highly confident even in regions far from the data. In Figure 37.3b, the penalty only encourages smoothness (upper Lipschitzness), which improves uncertainty estimates in comparison with (a) but still present areas of low uncertainty far from the training distribution. In Figure 37.3c, applying a two-sided penalty that enforces bi-Lipschitzness yields significantly better uncertainty estimates.

Although the two-sided gradient penalty appears effective in the example above, it also has important limitations. Computing Jacobians is expensive, and prior work has often relied on simplifications, such as using gradients of summary statistics (e.g., the gradient of the sum of kernel distances instead of the full Jacobian of the kernel distance vector). This difficulty restricts the method to using soft constraints, as in Equation 37.9, which not necessarily enforces bi-Lipschitzness. Furthermore, such soft constraints can introduce unintended side effects. For instance, causing residual connections to degenerate into identity mappings [1061].

37.4.2 Bi-Lipschitzness via Spectral Normalisation and Residual Connections

A faster alternative to Jacobian penalties for enforcing bi-Lipschitzness involves employing residual connections in the deep network and applying spectral normalisation to them. A residual connection is a neural network design where each layer learns only a residual mapping rather than a full transformation. Formally, for an input $\mathbf{x}$, a residual block is defined as $g_i(\mathbf{x}) = \mathbf{x} + h_i(\mathbf{x})$, where h_i is a learnable function (e.g., a neural network block). The term $\mathbf{x}$ is passed forward unchanged through a skip connection, while $h_i(\mathbf{x})$ models the residual. This simple addition helps stabilise optimisation, preserves distances under mild conditions, and enables training of very deep networks. Modern deep learning architectures are often composed of residual blocks, taking the form $g_{\theta_b}(\mathbf{x}) = g_{l-1} \circ g_{l-2} \circ \dots \circ g_1(\mathbf{x})$, with $l-1$ referring to the number of residual blocks.

One can show that if all residual functions are α -Lipschitz, for $0 < \alpha \leq 1$, i.e.:

$$\|h_i(\mathbf{x}) - h_i(\mathbf{x}')\| \leq \alpha \|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}', \quad (37.10)$$

then g_{θ_b} is also bi-Lipschitz with $L_1 = (1 - \alpha)^{l-1}$ and $L_2 = (1 + \alpha)^{l-1}$. Therefore, a sufficient condition is to ensure α -Lipschitzness in the residuals, which typically assume the form of $h_i = \sigma(\mathbf{W}_i \mathbf{x} + \mathbf{b}_i)$, where σ is a non-linear activation function (assumed 1-Lipschitz, e.g. ReLU or tanh), and $\mathbf{W}_i$ and $\mathbf{b}_i$ is the weight matrix and bias vector of the residual block i . We can achieve this by enforcing $\|\mathbf{W}_i\|_2 \leq 1$, since that:

$$\begin{aligned} \alpha &= \sup_{\mathbf{x} \neq \mathbf{x}'} \frac{\|h_i(\mathbf{x}) - h_i(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}} = \sup_{\mathbf{x} \neq \mathbf{x}'} \frac{\|\sigma(\mathbf{W}_i \mathbf{x} + \mathbf{b}_i) - \sigma(\mathbf{W}_i \mathbf{x}' + \mathbf{b}_i)\|}{\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}} \\ &\leq \sup_{\mathbf{x} \neq \mathbf{x}'} \frac{\|\mathbf{W}_i(\mathbf{x} - \mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}} = \|\mathbf{W}_i\|_2 \leq 1. \end{aligned} \quad (37.11)$$

The task, then, is to ensure that the spectral norm of each weight matrix $\mathbf{W}_i$ is properly controlled. We estimate this spectral norm using the power iteration method and apply the following normalisation:

$$\mathbf{W}_i = \begin{cases} c \cdot \mathbf{W}_i / \hat{\lambda}, & \text{if } \hat{\lambda} > c, \\ \mathbf{W}_i, & \text{otherwise,} \end{cases} \quad (37.12)$$

where $\hat{\lambda}$ is the estimated spectral norm and c is a hyperparameter that sets the desired upper bound. Since other architectural components can also influence the overall Lipschitz constant α , in practice we adjust c to ensure both smoothness and sensitivity.

Figure 37.4 illustrates the effect of spectral normalisation in enforcing bi-Lipschitzness for a Gaussian Process with a deep backbone. In this case, lighter shading indicates regions of high uncertainty. Without spectral normalisation (Figure 37.4a), the model remains overconfident even far from the training data (black and white points) and near the out-of-distribution dataset (the far grey blob of points). By contrast, applying spectral normalisation leads the deep GP to produce more reliable uncertainty estimates, both around the training distribution and in OOD regions.

Finally, Table 37.2 compares the methods discussed in this section using experiments from Liu et al. [1061]. All approaches share the same ResNet backbone trained on CIFAR-10, and evaluation is performed both in-distribution (accuracy on the CIFAR-10 test set) and out-of-distribution (Area Under the Precision-Recall Curve, AUPR, on SVHN and CIFAR-100). DUQ [1653] denotes the RBF network with a double-sided gradient penalty (subsection 37.4.1), DNN-SN applies spectral normalisation without a predictive head, and Spectral-Normalised Neural Gaussian Process (SNGP) [1061] combines spectral normalisation with a GP head. Observe that DNN-SN outperforms DUQ in-distribution and matches its OOD performance, while achieving much lower latency. Adding a Bayesian predictive head (SNGP) further improves OOD performance, achieving the best overall results.

Overall, spectral normalisation offers a simple but effective way to enforce bi-Lipschitzness, but it also presents notable limitations. It requires architectures with residual blocks. Furthermore, the condition in Equation 37.11 assumes dense residual functions, and different inductive biases (e.g., attention, convolution) may lead to different conditions. Other regularisation techniques (e.g., batch normalisation, dropout) also affect the Lipschitz constant, requiring a careful tuning of the hyperparameter c . Lastly, spectral normalisation is still costly, requiring repeated matrix-vector products for each residual block, which results in a substantial latency gap between a deterministic and a spectral normalised network, as shown in Table 37.2.

37.5 The predictive head

As stated in Section 37.2, we define the predictive head as a parameterised functional mapping $\mathcal{U}_{\theta_h} : \mathbb{R}^p \rightarrow \mathcal{P}(\mathcal{Y})$, which takes as input the features learned by the deep backbone and maps them to a distribution over outputs. Its

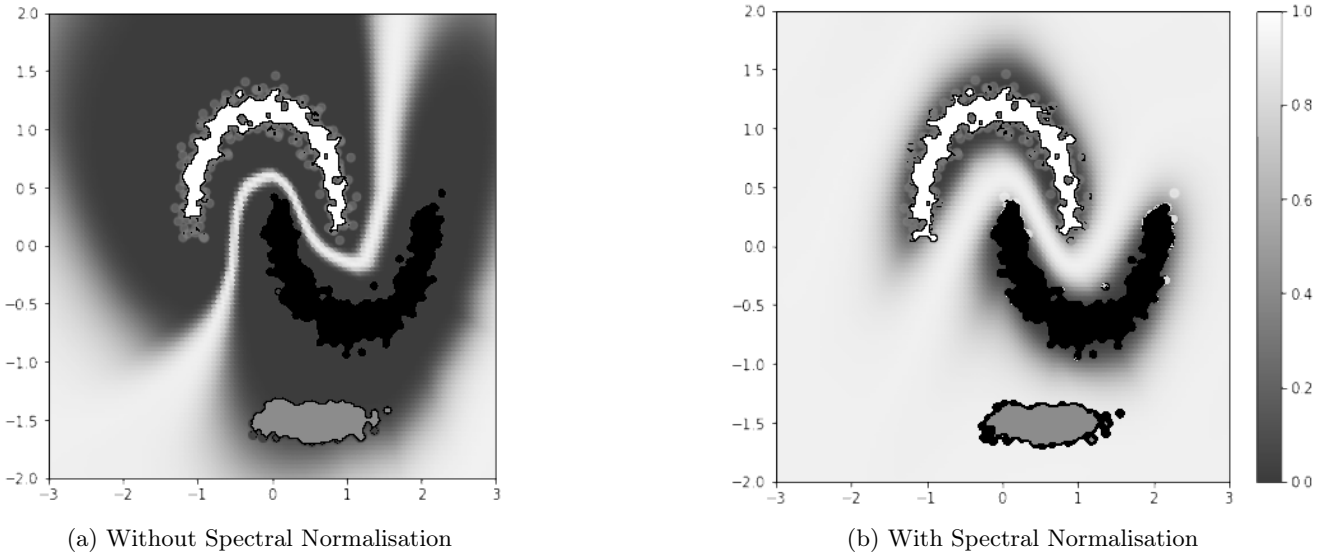


Figure 37.4: Uncertainty quantification on the two moons dataset using a Gaussian Process with a deep backbone network (ResNet). Black and white points denote positive and negative classes, while the far grey blob of points indicate OOD samples. Lighter shading indicates high uncertainty regions. Ideally, the model should be uncertain far from the training data. In (a), without enforcing bi-Lipschitzness, the model is overconfident in out-of-distribution regions. In (b), spectral normalisation enforces bi-Lipschitzness and yields more reliable uncertainty estimates. Adapted from Liu et al. [1061].

Method	Accuracy (ID) CIFAR-10	OOD AUPR ($\uparrow$)		Latency ($\downarrow$) (ms / example)
		SVHN	CIFAR-100	
Deterministic	96.0 ± 0.01	0.781 ± 0.01	0.835 ± 0.01	3.91
DUQ ¹	94.7 ± 0.02	0.973 ± 0.01	0.854 ± 0.01	8.68
DNN-SN	96.0 ± 0.01	0.974 ± 0.01	0.859 ± 0.01	5.20
SNGP	95.9 ± 0.01	0.990 ± 0.01	0.905 ± 0.01	6.25

Table 37.2: Comparison of gradient-penalty (DUQ) and spectral-normalisation approaches (DNN-SN, SNGP) on CIFAR-10 with a ResNet backbone. Spectral normalisation (DNN-SN) achieves higher in-distribution accuracy and similar OOD performance to the gradient-penalty method (DUQ) while running with much lower latency. Adding a Bayesian predictive head (SNGP) further boosts OOD performance. Adapted from Liu et al. [1061].

primary purpose is to provide reliable predictions and uncertainty estimates. In this section, we present a range of approaches for modelling this mapping, starting from deterministic methods and progressing to scalable Bayesian approximations.

37.5.1 The standard neural network head

The simplest way to represent the mapping $\mathcal{U}_{\theta_h}$ is with a deterministic neural network. For example, one might use a few-layers dense network with nonlinear activations (e.g., ReLU) in the hidden layers, and either a linear output layer (for Gaussian likelihoods) or a softmax output (for categorical likelihoods). When combined with the backbone, this forms a single deep network capable of producing predictions over $\mathcal{Y}$. But can we *trust* these predictions?

A crucial requirement for reliable predictions is that the distribution over $\mathcal{Y}$ be meaningful—that is, it should reflect the model’s uncertainty about the predicted variable. In other words, the model should *know what it does not know*, and express this through its predictive distribution. Standard neural networks, however, lack an inductive bias that enforces this property, and they can easily become overconfident on inputs far from the training data—precisely where we would expect low confidence. A simple illustration is a ReLU network, which one can show that it can produce outputs that grow without bound, leading to overconfident predictions. This pathology motivates the development of alternative heads that are uncertainty-aware by design.

37.5.2 The Radial Basis Function (RBF) head

A simple inductive bias that promotes uncertainty awareness is to explicitly model distances between input data and the training distribution in a representation space. Intuitively, if a point lies close to the training distribution, we expect its predictions to be consistent with those of nearby training examples, and the associated uncertainty should be low. Conversely, if a point is far from the training distribution in feature space, the model should express high uncertainty, as such inputs fall outside its domain of knowledge.

This concept is explicitly incorporated in *Radial Basis Function (RBF) networks*. These models construct predictions by measuring the similarity between an input and a set of reference points (often derived from the training data) in a feature space, where similarity is quantified in terms of Euclidean distances. Each reference point serves as the centre of a local receptive field, and the contribution of each centre to the final prediction decays with distance from the input. Formally, let $\{c_j \in \mathbb{R}^p : j = 1, \dots, m\}$ denote a collection of centres, and let $\phi : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}_{\geq 0}$ be a radial basis function, i.e. a function of the form

$$\phi(\mathbf{x}, c_j) = \varphi(\|g_{\theta_b}(\mathbf{x}) - c_j\|_2), \quad (37.13)$$

where $\varphi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is nonincreasing. A common choice is the Gaussian kernel,

$$\phi(\mathbf{x}, c_j) = \exp\left(-\frac{1}{2\sigma^2}\|g_{\theta_b}(\mathbf{x}) - c_j\|_2^2\right). \quad (37.14)$$

An RBF network defines a mapping $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$ of the form:

$$f(x) = \sum_{j=1}^m w_j \phi(\mathbf{x}, c_j), \quad (37.15)$$

where $w_j \in \mathbb{R}^q$ are trainable output weights. Thus, the prediction at x is expressed as a linear combination of local basis functions centred at the c_j . For classification tasks, the output of this linear combination is typically converted into class predictions either by applying a softmax to obtain probabilities or, in a prototype-based variant, by selecting the class whose centroid is closest to the input. The network is “local” in the sense that only the centres close to x make a significant contribution, explicitly encoding the principle that uncertainty grows as we move farther from the training distribution.

RBF networks have been used in prior work to quantify uncertainty in deep networks with a single forward pass, for example in DUQ [1653]. Nonetheless, these models are not truly probabilistic, and their notion of uncertainty is only implicit. Once the centres and weights are fixed, predictions reduce to deterministic point estimates. More importantly, they cannot disentangle epistemic from aleatoric uncertainty, which makes them less suitable for downstream tasks such as (Bayesian) active learning and experimental design.

37.5.3 The probabilistic head via generative classifiers

In the previous subsections, we described *discriminative heads*, which directly model $p(\mathbf{y} \mid \mathbf{x}, \theta_h)$. While discriminative classifiers are often effective for classification tasks, a key drawback is that they do not explicitly model the input distribution $p(\mathbf{x})$. This can lead to misleading uncertainty estimates: for example, in an RBF network that assigns labels by choosing the nearest centroid, the distance to that centroid can be small (suggesting low uncertainty) even if the point lies in a region where the true class has very low probability under its distribution. In other words, distance alone does not capture how concentrated or diffuse each class is.

An alternative is to use *generative classifiers*, which model the joint distribution $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{y})p(\mathbf{y})$ and recover predictions via Bayes’ rule. By capturing the full class-conditional densities $p(\mathbf{x} \mid \mathbf{y})$ along with the prior over labels, generative classifiers can account for class-specific variance and better distinguish between regions of high and low support in the input space. This enables them to assign low confidence to points that are far from high-density regions of *any* class, resulting in qualitatively different and more reliable uncertainty estimates.

The simplest and most widely studied example of a generative classifier is *Gaussian Discriminant Analysis (GDA)*. In GDA, the class-conditional densities are modelled as multivariate normal distributions,

$$p(\mathbf{x} \mid \mathbf{y} = c) = \mathcal{N}(g_{\theta_b}(\mathbf{x}); \mu_c, \Sigma_c), \quad (37.16)$$

and the prior over classes is often chosen to be Bernoulli in the binary case, or categorical in the multi-class case, i.e., $P(\mathbf{y} = c) = \frac{\beta_c}{\sum_{c'} \beta_{c'}}$, where β_c is the unnormalised prior for class c . A particularly interesting setting arises

when we assume the covariances are tied across classes: $\Sigma_c = \Sigma$ for all c . This corresponds to the classical *Linear Discriminant Analysis (LDA)* model. In this case, the posterior takes the following form:

$$P(\mathbf{y} = c \mid \mathbf{x}) = \frac{P(\mathbf{y} = c) p(\mathbf{x} \mid \mathbf{y} = c)}{\sum_{c'} P(\mathbf{y} = c') p(\mathbf{x} \mid \mathbf{y} = c')} = \frac{\exp(\mu_c^\top \Sigma^{-1} g_{\theta_b}(\mathbf{x}) - \frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \log \beta_c)}{\sum_{c'} \exp(\mu_{c'}^\top \Sigma^{-1} g_{\theta_b}(\mathbf{x}) - \frac{1}{2} \mu_{c'}^\top \Sigma^{-1} \mu_{c'} + \log \beta_{c'})}. \quad (37.17)$$

One can immediately recognise that this has the same form as a softmax classifier: the term $\mu_c^\top \Sigma^{-1}$ plays the role of the weight vector, while $-\frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \log \beta_c$ corresponds to a bias. This equivalence shows that training a softmax classifier implicitly corresponds to fitting a Gaussian model of the data, and moreover, that in principle we may recover the underlying densities from the discriminative formulation. To do so, we may compute the empirical class means and the tied covariance matrix as:

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i: y_i = c} g_{\theta_b}(\mathbf{x}_i), \quad \hat{\Sigma} = \frac{1}{N} \sum_c \sum_{i: y_i = c} (g_{\theta_b}(\mathbf{x}_i) - \hat{\mu}_c)(g_{\theta_b}(\mathbf{x}_i) - \hat{\mu}_c)^\top, \quad (37.18)$$

where N_c is the number of training samples with label c . This procedure is equivalent to fitting the class-conditional Gaussian distributions with a tied covariance to the training features under the maximum likelihood estimator. With the class-conditional distributions estimated, prediction for a new input x reduces to measuring its distance from each class. A natural choice is the *Mahalanobis distance*,

$$D_c(\mathbf{x}) = \sqrt{(g_{\theta_b}(\mathbf{x}) - \hat{\mu}_c)^\top \hat{\Sigma}^{-1} (g_{\theta_b}(\mathbf{x}) - \hat{\mu}_c)}, \quad (37.19)$$

which accounts for the covariance structure of the data. The predicted label is the class with the smallest distance, while the distance itself serves as an uncertainty measure: small values indicate high density under class c , and large values correspond to low support, often signalling out-of-distribution inputs [1004]. Alternatively, we may directly evaluate the marginal likelihood of the feature representation as an uncertainty score:

$$p(\mathbf{x}) = \sum_c p(\mathbf{x} \mid \mathbf{y} = c) p(\mathbf{y} = c). \quad (37.20)$$

This principle is used in Deep Deterministic Uncertainty (DDU) [1206]. DDU directly fits a feature-density estimator using GDA without assuming tied covariances. It fits a single Gaussian per class by computing the empirical mean and covariance of the feature vectors for each of them. Then, it estimates epistemic uncertainty via Equation 37.20.

So far, the methods presented for functional heads only provide *proxies* for uncertainty. These are heuristics that resemble uncertainty estimates, as they capture the intuition that points far from the training distribution should be assigned higher uncertainty. However, they do not account for the fundamental source of epistemic uncertainty: our limited knowledge about the model parameter, which is formally captured via Bayesian posteriors. Next, we focus on truly Bayesian heads via Gaussian Processes, and discuss scalable methods for approximate inference.

37.5.4 The Bayesian head via Gaussian Processes

A natural progression from previous methods is to move toward a fully Bayesian treatment of function uncertainty. Gaussian Processes (GPs), covered in more depth in Chapter 23 and in the scalable-GP treatment of Chapter 38, provide such a framework: they define a prior over functions, and yield a posterior distribution that quantifies uncertainty in a principled way. GPs can be seen as a Bayesian analogue of kernel methods, where instead of heuristically measuring distances or fitting Gaussian densities in feature space, we directly model distributions over functions that explain the data.

Formally, a Gaussian Process is a distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$ such that any finite collection of function values is jointly Gaussian,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad (37.21)$$

where $m(\mathbf{x})$ is the mean function (often set to zero) and $k(\mathbf{x}, \mathbf{x}')$ is the covariance kernel encoding smoothness and similarity assumptions. Given training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ and assuming Gaussian observation noise, GP inference is analytically tractable. The predictive distribution at a new input $\mathbf{x}^*$ is Gaussian with mean and variance

$$\mu(\mathbf{x}^*) = k(\mathbf{x}^*, X) (K + \sigma^2 I)^{-1} \mathbf{y}, \quad \sigma^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, X) (K + \sigma^2 I)^{-1} k(X, \mathbf{x}^*), \quad (37.22)$$

where K is the kernel matrix over the training inputs. Thus, predictions are accompanied by an explicit measure of uncertainty: high variance is assigned to regions far from observed data, reflecting epistemic uncertainty. Despite their conceptual elegance, GPs face important limitations: exact inference scales as $\mathcal{O}(N^3)$ in the number of data points N due to the inversion of K , and non-Gaussian likelihoods (e.g. for classification) render the posterior analytically intractable. In the following subsections, we will explore approximations that make Gaussian Processes practical in large-scale and non-Gaussian settings.

Approximated GP Inference via Random Fourier Features

Since all GP computations depend on the kernel matrix K , whose inversion costs $\mathcal{O}(N^3)$, replacing $k(x, x')$ with a fast approximation can dramatically reduce the complexity. Random Fourier Features (RFF) [1363] provide such an approximation for stationary kernels. By Bochner's theorem [176], any stationary kernel can be expressed as the Fourier transform of a positive measure, in particular:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\omega \sim p} [\cos(\omega^\top (\mathbf{x} - \mathbf{x}'))], \quad (37.23)$$

where $p(\omega)$ is the spectral density of the kernel. Approximating this expectation with m Monte Carlo samples yields the feature map:

$$\phi(\mathbf{x}) = \sqrt{\frac{2}{m}} (\cos(\omega_1^\top \mathbf{x} + b_1), \dots, \cos(\omega_m^\top \mathbf{x} + b_m))^\top, \quad (37.24)$$

where $b_i \sim \text{Uniform}[0, 2\pi]$; the random phase b_i makes the estimator unbiased, since a product-to-sum expansion gives $\mathbb{E}_b[2 \cos(\omega^\top \mathbf{x} + b) \cos(\omega^\top \mathbf{x}' + b)] = \cos(\omega^\top (\mathbf{x} - \mathbf{x}'))$ (the $\cos(\omega^\top (\mathbf{x} + \mathbf{x}') + 2b)$ term integrates to zero over $b \sim \text{Uniform}[0, 2\pi]$). The kernel is then replaced by the inner product $k(\mathbf{x}, \mathbf{x}') \approx \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. With this approximation, the GP prior can be reformulated as a Bayesian linear model in the random feature space. We place a Gaussian prior on the weights, $\mathbf{w} \sim \mathcal{N}(0, I)$, and define the function as

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}. \quad (37.25)$$

For training inputs $X = \{\mathbf{x}_i\}_{i=1}^N$, stacking the features into $\Phi \in \mathbb{R}^{N \times m}$ gives

$$\mathbf{f} = \Phi \mathbf{w}, \quad \mathbf{f} \sim \mathcal{N}(0, \Phi \Phi^\top), \quad (37.26)$$

so that $\Phi \Phi^\top$ serves as a low-rank approximation to the exact kernel matrix K .

The computational advantage is immediate: instead of inverting the $N \times N$ kernel matrix, we only need to handle the $m \times m$ system arising from the weight posterior. This reduces the cost from $\mathcal{O}(N^3)$ to $\mathcal{O}(Nm^2 + m^3)$, with $m \ll N$. In practice, a few hundred features are often sufficient, making GP-style inference feasible even for datasets with tens of thousands of points. This is how SNGP [1061] enables BDL with a single forward pass in a deep backbone.

Nevertheless, a key limitation of RFF is that it sacrifices the non-parametric property of GPs. For any finite number of features, the GP becomes a parametric linear model in a fixed basis. As the dataset grows, the posterior variance can erroneously shrink to zero everywhere, even in regions far from the data, unlike a true GP which maintains uncertainty away from observations.

Approximated Inference via Variational Inducing Points

While Random Fourier Features approximate the kernel function itself, an alternative route is to retain the exact GP prior and instead approximate the posterior distribution in a scalable way [1615]. The key observation is that a GP is entirely specified by its values on any finite set of inputs, so one can summarise the function through a small number of carefully chosen pseudo-points. These inducing inputs provide a low-rank representation that can be optimised to best explain the data while maintaining consistency with the GP prior.

Concretely, let $Z = \{z_j\}_{j=1}^m$ be a set of inducing inputs with corresponding function values $\mathbf{u} = (f(z_1), \dots, f(z_m))^\top$. For training inputs $X = \{\mathbf{x}_i\}_{i=1}^N$, the joint GP prior over $\mathbf{f} = (f(x_1), \dots, f(x_N))^\top$ and $\mathbf{u}$ is Gaussian:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{ss} & K_{sm} \\ K_{ms} & K_{mm} \end{bmatrix}\right), \quad (37.27)$$

where $K_{ss} = K_{XX}$, $K_{mm} = K_{ZZ}$, and $K_{sm} = K_{XZ}$. From standard Gaussian conditioning, the conditional distribution of $\mathbf{f}$ given $\mathbf{u}$ is:

$$p(\mathbf{f} | \mathbf{u}) = \mathcal{N}(K_{sm} K_{mm}^{-1} \mathbf{u}, K_{ss} - Q_{ss}), \quad (37.28)$$

with $Q_{ss} = K_{sm}K_{mm}^{-1}K_{ms}$. Once we know the function values at the inducing points, the rest of the process is Gaussian with mean given by interpolation through Z and residual covariance measuring what cannot be explained by that interpolation.

The goal is to approximate the posterior over $\mathbf{f}$ given observations $\mathbf{y}$. The marginal likelihood of the model is:

$$p(\mathbf{y}) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \mathbf{u}) p(\mathbf{u}) d\mathbf{f} d\mathbf{u}, \quad (37.29)$$

but this integral is generally intractable. To address this, we introduce a variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$ and define:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} | \mathbf{u}) q(\mathbf{u}). \quad (37.30)$$

This preserves the exact conditional structure of the GP and only approximates the distribution over inducing variables. Applying Jensen’s inequality yields the evidence lower bound (ELBO):

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y} | \mathbf{f})] - D_{\text{KL}}(q(\mathbf{u}) \| p(\mathbf{u})), \quad (37.31)$$

where $q(\mathbf{f}) = \int p(\mathbf{f} | \mathbf{u}) q(\mathbf{u}) d\mathbf{u}$ is Gaussian with mean $K_{sm}K_{mm}^{-1}\mathbf{m}$ and covariance $K_{ss} + K_{sm}K_{mm}^{-1}(\mathbf{S} - K_{mm})K_{mm}^{-1}K_{ms}$. This objective can be maximised with respect to the variational parameters $(\mathbf{m}, \mathbf{S})$, the inducing locations Z , and the kernel hyperparameters.

For Gaussian likelihoods, this bound can be evaluated in closed form, leading to a particularly elegant expression in which the log marginal likelihood of the Nyström approximation Q_{ss} is corrected by a trace term that penalises information lost outside the inducing subspace. For general likelihoods, the bound remains tractable through Monte Carlo estimates, and its factorised structure allows the use of minibatches. This makes the approach suitable for very large datasets.

The computational benefits are similar to those of RFFs: instead of inverting the full $N \times N$ kernel matrix, training reduces to operations involving the $m \times m$ covariance among the inducing points. This reduces complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(Nm^2 + m^3)$, with m typically in the hundreds. Unlike RFFs, however, the basis here is *not fixed in advance*. The inducing locations are variational parameters learned to maximize the evidence bound, adapting flexibly to the data distribution. Moreover, because the exact conditional structure of the GP is preserved, the predictive distribution retains good uncertainty estimation, with the trace correction ensuring that posterior variance does not collapse spuriously far from the data. Deterministic Uncertainty Estimation (DUE) [1654] uses this GP approximation method for BDL with a single forward pass.

Nevertheless, inducing point methods also have clear limitations. The quality of the approximation is bounded by the span of the inducing variables: if m is too small relative to the complexity of the function, the variational family may be too restrictive, leading to underfitting. Although increasing m makes the method arbitrarily close to the exact GP, this comes at the cost of higher $\mathcal{O}(m^3)$ operations, which can again become prohibitive. Optimising the inducing locations jointly with kernel hyperparameters is a non-convex problem and can be sensitive to initialisation. Finally, in very high-dimensional input spaces, it can be challenging to place inducing points effectively, since covering the relevant regions of the input space requires exponentially many points.

37.6 Outlook

This chapter introduced several methods that form the foundation of scalable BDL with a single forward pass. We now compare these approaches quantitatively to highlight how the different predictive heads and backbone regularisation schemes trade off predictive performance, uncertainty estimation, and computational efficiency.

Table 37.3 summarizes the performance of a range of methods that require only a single forward pass at inference, along with a standard deterministic neural network and Deep Ensembles. All models were trained on CIFAR-10 and evaluated both in-distribution (on the CIFAR-10 test set) and out-of-distribution (on SVHN). The results are adapted from Mukhoti et al. [1206], van Amersfoort et al. [1654].

Deep Ensembles deliver the strongest overall performance in both accuracy and uncertainty calibration but at a much higher computational cost—roughly five times that of a single model. They therefore serve as a useful reference point for assessing more efficient alternatives that aim to approach ensemble-level reliability at lower cost.

Among single-pass models, those employing spectral normalisation to enforce approximate bi-Lipschitzness, such as DDU and SNGP, perform best. Spectral normalisation provides an effective regularisation mechanism: it stabilises the learned representation and constrains the sensitivity of the predictive head, while keeping runtime roughly similar to that of a standard network. In contrast, Jacobian regularisation (as in DUQ) or the absence of any constraint tends to yield less stable uncertainty estimates and weaker predictive performance.

Method	Regulariser	Head	Runtime	Accuracy ($\uparrow$)	AUROC ² SVHN ($\uparrow$)
Deterministic NN	-	Standard NN	1x	95.98 $\pm$ 0.02	94.44 $\pm$ 0.43
DUQ [1653]	Jacobian	RBF Network	3.5x	94.6 $\pm$ 0.16	93.71 $\pm$ 0.61
DDU [1206]	Spectral	GDA	1x	95.97 $\pm$ 0.03	97.86 $\pm$ 0.19
SNGP [1061]	Spectral	GP (RFFs)	1x	96.04 $\pm$ 0.09	94.0 $\pm$ 1.3
DUE [1654]	Spectral	GP (Inducing)	1x	95.6 $\pm$ 0.04	95.8 $\pm$ 0.005
Deep Ensembles [962]	-	Standard NN	5x	96.59 $\pm$ 0.02	97.73 $\pm$ 0.31

Table 37.3: Comparison of scalable Bayesian deep learning methods on CIFAR-10 (in-distribution) and SVHN (out-of-distribution). Spectral normalisation-based approaches (DDU, SNGP) achieve the best trade-off between predictive performance, uncertainty calibration, and efficiency, outperforming Jacobian-regularised and unregularised baselines. While Deep Ensembles remain the strongest overall, they do so at a substantially higher computational cost, highlighting the progress and remaining challenges in developing scalable single-pass uncertainty estimators. Adapted from Mukhoti et al. [1206] and van Amersfoort et al. [1654].

Within this family, DDU achieves the best out-of-distribution evaluation and matches SNGP in in-distribution accuracy. Fully Bayesian approaches—such as Gaussian Process heads with Random Fourier Features or inducing points—improve upon deterministic networks but still fall short of DDU in both accuracy and calibration. This gap suggests that, while Bayesian formulations provide a principled foundation for uncertainty estimation, practical implementations remain limited by computational constraints. Even with low-rank approximations, training and inference scale cubically with the number of features or inducing points, restricting the achievable model capacity.

Overall, the methods covered in this chapter have addressed key challenges in combining deep learning with Bayesian principles, leading to models capable of producing reliable, uncertainty-aware predictions. Nevertheless, there is still room for improvement, particularly for fully Bayesian predictive heads, to achieve state-of-the-art predictive power and uncertainty estimation at a reasonable computational cost.

Chapter 38

Scalable Gaussian processes

Gaussian processes [GPs; 1382], as introduced in Chapter 26, are gold standard methods for function space uncertainty quantification. Although they typically achieve accurate predictions with reliable uncertainty estimates on small datasets and relatively low-dimensional problems, it remains a challenge to scale them up to big datasets with high-dimensional inputs. The issue of the curse of dimensionality might be mitigated with better kernel design, e.g., deep kernel learning (introduced in Chapter 24), and we will not discuss it in detail in this chapter since it is prevalent in all kernel methods. Instead, we focus on reviewing methods that scale GPs to big data with a large number of data points. The main caveat of applying full GP in big data is the requirement of computing matrix inversion and matrix determinant involving the kernel matrix, $\mathbf{K}_{\mathbf{X}\mathbf{X}}$, evaluated over all training inputs, $\mathbf{X}$, which scales cubically w.r.t the number of training instances. Therefore, approximations that bypass matrix inversion and matrix determinant computation for large matrices are indispensable.

In this chapter, we review two classes of structured approximation techniques commonly used in the context of GPs. The first class of approaches directly employ low-rank approximations for $\mathbf{K}_{\mathbf{X}\mathbf{X}}$, and among them, Nyström approximation [1760] and random Fourier feature (RFF) [1363] are two prominent techniques. The second class of approaches, variational inducing points based approximations, are more tailored to GP models and we will discuss them in more detail. Specifically, a sparse variational Gaussian process [SVGP; 1528] constructs an approximate posterior process in a variational inference (VI) framework [851] (introduced in Chapter 12), and this approximate posterior is a structured GP fully determined by a small number of tunable inducing points, which only requires matrix inversion and determinant computation for the kernel matrix evaluated over the inducing points instead of the full set of training inputs. We also discuss decoupled SVGPs [285, 1454, 1513] and interdomain GPs [1663] as extensions of this framework. Furthermore, for streaming tasks where new data points keep being observed, storing all past data for retraining is expensive, and we resort to online or continual GP methods [218, 278] to bypass the requirement of storing all the data for retraining, but also to allow the model to maintain consistent performance across all the past tasks.

38.1 Direct approximations for $\mathbf{K}_{\mathbf{X}\mathbf{X}}$

In Chapter 23, we see that the main computational cost of posterior inference in GP regression comes from the matrix inversion $(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2 \mathbf{I})^{-1}$ in the posterior predictive distribution, and the log-determinant evaluation $\log \det(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2 \mathbf{I})$ in the log-marginal-likelihood objective for tuning the kernel hyperparameters. Low-rank matrix approximation can be adopted to address this issue. Suppose $\mathbf{K}_{\mathbf{X}\mathbf{X}} \in \mathbb{R}^{N \times N}$ admits a structured approximation of the form $\mathbf{K}_{\mathbf{X}\mathbf{X}} \approx \Phi^\top \Sigma \Phi$, where $\Phi \in \mathbb{R}^{M \times N}$, $\Sigma \in \mathbb{R}^{M \times M}$ and $M \ll N$. Then, by applying matrix inversion and matrix determinant calculations to the low-rank approximation, the original matrix inverse and determinant terms, which require $O(N^3)$ time complexity in computation, can be approximately computed with $O(NM^2 + M^3)$ time complexity:

$$(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2 \mathbf{I})^{-1} \approx \frac{1}{\sigma^2} \mathbf{I} - \frac{1}{\sigma^4} \Phi^\top (\Sigma^{-1} + \frac{1}{\sigma^2} \Phi \Phi^\top)^{-1} \Phi, \quad (38.1)$$

$$\log \det(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2 \mathbf{I}) \approx 2N \log \sigma + \log \det(\Sigma^{-1} + \frac{1}{\sigma^2} \Phi \Phi^\top) + \log \det(\Sigma). \quad (38.2)$$

A few popular kernel approximation methods employ the structured form introduced above [1027, 1322, 1363, 1623, 1760], and we review Nyström approximation [1760] and random Fourier feature approximation [1363] as two examples.

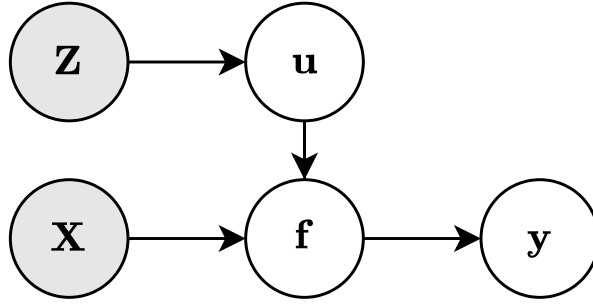


Figure 38.1: The graphical model for prediction with a sparse variational Gaussian process (SVGP). Shaded nodes ($\mathbf{X}$, $\mathbf{y}$) are observed and unshaded nodes are latent. Here $\mathbf{Z}$ denotes the inducing input locations (treated as hyperparameters), $\mathbf{u}$ the corresponding inducing function values, $\mathbf{f}$ the latent function values at the training inputs $\mathbf{X}$, and $\mathbf{y}$ the observed targets.

Nystrom approximation. Nystrom approximation [1760], which is closely related to the subset-of-regressors, DTC, and FITC approximations unified in Quiñero-Candela and Rasmussen [1354], considers Φ to be $\mathbf{K}_{\mathbf{Z}\mathbf{X}}$ and Σ to be $\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}$, where $\mathbf{Z} \subset \mathbf{X}$ is a subset of M inputs randomly selected from the training set. Consequently, the kernel matrix evaluated on the training set is approximated as $\mathbf{K}_{\mathbf{X}\mathbf{X}} \approx \mathbf{K}_{\mathbf{X}\mathbf{Z}}\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z}\mathbf{X}}$, and the matrix inverse and log-determinant terms are approximated as follows:

$$(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I})^{-1} \approx \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\mathbf{K}_{\mathbf{X}\mathbf{Z}}(\mathbf{K}_{\mathbf{Z}\mathbf{Z}} + \frac{1}{\sigma^2}\mathbf{K}_{\mathbf{Z}\mathbf{X}}\mathbf{K}_{\mathbf{X}\mathbf{Z}})^{-1}\mathbf{K}_{\mathbf{Z}\mathbf{X}}, \quad (38.3)$$

$$\log \det(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I}) \approx 2N \log \sigma + \log \det(\mathbf{K}_{\mathbf{Z}\mathbf{Z}} + \frac{1}{\sigma^2}\mathbf{K}_{\mathbf{Z}\mathbf{X}}\mathbf{K}_{\mathbf{X}\mathbf{Z}}) - \log \det(\mathbf{K}_{\mathbf{Z}\mathbf{Z}}). \quad (38.4)$$

Random Fourier features. For stationary kernels (i.e., $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$), Bochner's theorem [1433] provides a Fourier feature approximation:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &\approx \frac{1}{M} \sum_{m=1}^M [\cos(\mathbf{w}_m^\top \mathbf{x}) \cos(\mathbf{w}_m^\top \mathbf{x}') + \sin(\mathbf{w}_m^\top \mathbf{x}) \sin(\mathbf{w}_m^\top \mathbf{x}')] \\ &= \phi(\mathbf{x})^\top \phi(\mathbf{x}'), \quad \mathbf{w}_m \sim p(\mathbf{w}), \end{aligned} \quad (38.5)$$

where $\phi(\mathbf{x}) = [\cos(\mathbf{w}_1^\top \mathbf{x}), \dots, \cos(\mathbf{w}_M^\top \mathbf{x}), \sin(\mathbf{w}_1^\top \mathbf{x}), \dots, \sin(\mathbf{w}_M^\top \mathbf{x})]^\top$ and $p(\mathbf{w})$ is the spectral density of the stationary kernel. Based on this result, given M samples from the spectral density, $\{\mathbf{w}_m\}_{m=1}^M$, random Fourier feature (RFF) approach [1363] approximates $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ as $\mathbf{K}_{\mathbf{X}\mathbf{X}} \approx \Phi^\top \Phi$, where $\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)]^\top$. Again, RFF approximation admits the desired structured form by simply noticing that $\Sigma = \mathbf{I}$ here. As a result, the matrix inversion and log-determinant terms are approximated as follows:

$$(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I})^{-1} \approx \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\Phi^\top (\mathbf{I} + \frac{1}{\sigma^2}\Phi\Phi^\top)^{-1}\Phi \quad (38.6)$$

$$\log \det(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I}) \approx 2N \log \sigma + \log \det(\mathbf{I} + \frac{1}{\sigma^2}\Phi\Phi^\top). \quad (38.7)$$

For non-stationary kernels, more advanced Fourier feature techniques (e.g., Ton et al. [1623]) may be leveraged to construct the kernel approximation. Moreover, there exist Fourier features tailored to some specific kernels, such as the RBF/Gaussian kernel, that achieve more accurate approximations for high-dimensional inputs [1387, 1822].

38.2 Sparse variational Gaussian processes

This section reviews sparse variational Gaussian process [1528], a VI framework that extends GPs to big data. SVGP considers a structured GP as the approximate posterior process and its structure is fully determined by a small set (with size $M \ll N$) of pseudo input-function value pairs, $\{(\mathbf{z}_m, u_m := f(\mathbf{z}_m))\}_{m=1}^M$, which are commonly known as inducing points [737, 738, 1615].

With the additional inducing points, the prior model for the augmented set consisting of $\mathbf{u}$ and $\mathbf{f}$ (evaluated at arbitrary inputs $\mathbf{X}$), according to the prior GP $p(f) = \mathcal{GP}(0, k_\psi(\cdot, \cdot))$ with kernel hyperparameters ψ , is

$$p(\mathbf{f}, \mathbf{u} | \mathbf{X}, \mathbf{Z}) = \mathcal{N}(\mathbf{0}, \begin{pmatrix} \mathbf{K}_{\mathbf{XX}} & \mathbf{K}_{\mathbf{XZ}} \\ \mathbf{K}_{\mathbf{ZX}} & \mathbf{K}_{\mathbf{ZZ}} \end{pmatrix}). \quad (38.8)$$

SVGP then constructs an approximate posterior GP based on a Gaussian variational distribution for the set of inducing points, $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}_{\mathbf{u}}, \mathbf{S}_{\mathbf{u}})$, where $\mathbf{m}_{\mathbf{u}}$ and $\mathbf{S}_{\mathbf{u}}$ are tunable variational mean and variational covariance parameters. Furthermore, we assume that the conditional distribution of function values given inducing variables in this approximate posterior is the same as the prior conditional distribution.

$$(\text{prior conditional matching}) \quad q(\mathbf{f} | \mathbf{u}, \mathbf{Z}, \mathbf{X}) = p(\mathbf{f} | \mathbf{u}, \mathbf{Z}, \mathbf{X}). \quad (38.9)$$

With this prior conditional matching assumption, the induced approximate posterior for function values $\mathbf{f}$ is completely controlled by the variational distribution $q(\mathbf{u})$, and it can be shown to follow a structured Gaussian distribution

$$\begin{aligned} q(\mathbf{f} | \mathbf{X}, \mathbf{Z}) &= \int p(\mathbf{f} | \mathbf{u}, \mathbf{Z}, \mathbf{X}) q(\mathbf{u}) d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f}; \mathbf{K}_{\mathbf{XZ}} \mathbf{K}_{\mathbf{ZZ}}^{-1} \mathbf{m}_{\mathbf{u}}, \mathbf{K}_{\mathbf{XX}} + \mathbf{K}_{\mathbf{XZ}} \mathbf{K}_{\mathbf{ZZ}}^{-1} (\mathbf{S}_{\mathbf{u}} - \mathbf{K}_{\mathbf{ZZ}}) \mathbf{K}_{\mathbf{ZZ}}^{-1} \mathbf{K}_{\mathbf{ZX}}). \end{aligned} \quad (38.10)$$

Clearly, when $\mathbf{X}$ is the training set, the approximate posterior above reduces the $O(N^3)$ time complexity of full GP to $O(M^3 + NM^2)$ by noticing that the expensive matrix inversion is now applied to a smaller matrix $\mathbf{K}_{\mathbf{ZZ}}$ (with $O(M^3)$ time complexity). The posterior predictive distribution of $\mathbf{y}$ is then obtained via the following integral

$$q(\mathbf{y} | \mathbf{X}, \mathbf{Z}) = \int \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \mathbf{u}, \mathbf{Z}, \mathbf{X}) q(\mathbf{u}) d\mathbf{u} d\mathbf{f} = \mathbb{E}_{q(\mathbf{f} | \mathbf{X}, \mathbf{Z})} [p(\mathbf{y} | \mathbf{f})]. \quad (38.11)$$

We plot the graphical model for the posterior prediction of SVGP in Figure 38.1. The flexibility of specifying tractable approximate posterior in the VI framework allows us to handle the cases where non-conjugate observation likelihoods are considered. Specifically, in the case of SVGP, although the posterior predictive distribution above has no closed form with non-Gaussian likelihoods, we can still use Monte Carlo to obtain an unbiased estimate since we can easily draw samples from $q(\mathbf{f} | \mathbf{X}, \mathbf{Z})$, which is a Gaussian (Eq. 38.10). In regression problems, where Gaussian observation likelihood is used, this posterior predictive distribution can be computed analytically as a Gaussian and the SVGP in this case is known as sparse Gaussian process regression [SGPR; 1615].

To ensure this approximate posterior to be close to the true posterior, we tune the inducing input locations $\mathbf{Z}$ (treated as model hyperparameters), the kernel hyperparameters and the variational parameters jointly by maximizing the evidence lower bound (ELBO). Specifically, with a training set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, the ELBO of SVGP is given as follows

$$\begin{aligned} \mathcal{L}_{ELBO} &= \mathbb{E}_{q(\mathbf{f}, \mathbf{u} | \mathbf{Z}, \mathbf{X})} \left[\log \frac{p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \mathbf{u}, \mathbf{Z}, \mathbf{X}) p(\mathbf{u} | \mathbf{Z})}{q(\mathbf{f}, \mathbf{u} | \mathbf{Z}, \mathbf{X})} \right] \\ &= \underbrace{\mathbb{E}_{q(\mathbf{f} | \mathbf{X}, \mathbf{Z})} [\log p(\mathbf{y} | \mathbf{f})]}_{\text{ELL}} - \underbrace{\text{KL}(q(\mathbf{u}) || p(\mathbf{u} | \mathbf{Z}))}_{\text{KL regulariser}} \end{aligned} \quad (38.12)$$

The above ELBO consists of two terms. The first term is commonly referred to as expected log-likelihood (ELL), which encourages better data fit, and the second term is a KL regulariser preventing the approximate posterior from significantly deviating from the prior. Since $q(\mathbf{u})$ and $p(\mathbf{u} | \mathbf{Z})$ are both Gaussian, the KL regulariser can be evaluated analytically as follows.

$$\text{KL}[q(\mathbf{u}) || p(\mathbf{u} | \mathbf{Z})] = \frac{1}{2} \mathbf{m}_{\mathbf{u}}^\top \mathbf{K}_{\mathbf{ZZ}}^{-1} \mathbf{m}_{\mathbf{u}} + \frac{1}{2} \text{Tr}[\mathbf{K}_{\mathbf{ZZ}}^{-1} \mathbf{S}_{\mathbf{u}} - \mathbf{I}] - \frac{1}{2} \log \det(\mathbf{K}_{\mathbf{ZZ}}^{-1} \mathbf{S}_{\mathbf{u}}). \quad (38.13)$$

Although the ELL term has no closed form with non-Gaussian likelihoods, we can again resort to Monte Carlo to estimate it. The main costs for the evaluation of the ELL and KL regulariser come from the computation of $q(\mathbf{f} | \mathbf{X}, \mathbf{Z})$ ($O(M^3 + NM^2)$), and the computation of the determinant and inversion of $\mathbf{K}_{\mathbf{ZZ}}$ ($O(M^3)$), respectively, which are significantly lower than the $O(N^3)$ time complexity of full GP regression.

In SGPR where Gaussian observation likelihood is in use, the ELL term can also be computed analytically and the optimal variational mean and covariance parameters in $q(\mathbf{u})$ can be derived in closed forms as functions of inducing locations $\mathbf{Z}$ and kernel hyperparameters. As a result, the ELBO can be simplified to a collapsed bound [1615] for training $\mathbf{Z}$ and kernel hyperparameters only. This collapsed bound is particularly useful in practice as it eliminates the variational parameters entirely, reducing the number of optimisable parameters and providing a tighter lower bound on the log-marginal likelihood.

38.3 Decoupled sparse variational Gaussian processes

While standard SVGP considers direct optimisation of the variational parameters (i.e., mean and covariance of $q(\mathbf{u})$) for the inducing points, decoupled SVGPs [285, 1454, 1513] further improve the efficiency of sparse GP inference by imposing structures in the variational parameters, allowing structured decoupling of the inducing points into two complementary subsets, used to construct the posterior predictive mean and covariance, respectively.

Consider splitting the inducing points to obtain two complementary subsets: $(\mathbf{Z}, \mathbf{u}) = \{(\mathbf{Z}_\gamma, \mathbf{u}_\gamma), (\mathbf{Z}_\beta, \mathbf{u}_\beta)\}$, where the index sets $\gamma, \beta \subset \{1, \dots, M\}$ satisfy $\gamma \cap \beta = \emptyset, \gamma \cup \beta = \{1, \dots, M\}$. Decoupled SVGP [DSVGP; 285] then considers a structured variational distribution for the inducing points according to the split as follows

$$q(\mathbf{u}) = q(\mathbf{u}_\gamma)q(\mathbf{u}_\beta|\mathbf{u}_\gamma) = \mathcal{N}(\mathbf{u}_\gamma; \mathbf{0}, \mathbf{S}_\gamma)\mathcal{N}(\mathbf{u}_\beta; \mathbf{C}\mathbf{u}_\gamma + \mathbf{m}_\beta, \mathbf{B}), \quad (38.14)$$

where $\mathbf{B} = \mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta} - \mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\gamma}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\beta}$ and $\mathbf{C} = \mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\gamma}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}$. The corresponding variational mean and covariance are:

$$\mathbf{m}_\mathbf{u} = \begin{pmatrix} \mathbf{0} \\ \mathbf{m}_\beta \end{pmatrix}, \quad \mathbf{S}_\mathbf{u} = \begin{pmatrix} \mathbf{S}_\gamma & \mathbf{S}_\gamma\mathbf{C}^\top \\ \mathbf{C}\mathbf{S}_\gamma & \mathbf{B} + \mathbf{C}\mathbf{S}_\gamma\mathbf{C}^\top \end{pmatrix}. \quad (38.15)$$

It can then be shown that:

$$\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}\mathbf{S}_\mathbf{u} = \begin{bmatrix} \mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma & (\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma - \mathbf{I})\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\beta} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad (38.16)$$

$$\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}\mathbf{S}_\mathbf{u}\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1} - \mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1} = \begin{bmatrix} \mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1} - \mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (38.17)$$

Integrating this new design of the variational mean and covariance parameters into the posterior predictive distribution of SVGP (Eq. 38.10) allows us to obtain the posterior distribution $q(\mathbf{f}|\mathbf{X}, \mathbf{Z})$ for DSVGP, which is a Gaussian with the following mean $\mathbf{m}_\mathbf{f}$ and covariance $\Sigma_\mathbf{ff}$:

$$\begin{aligned} \mathbf{m}_\mathbf{f} &= \mathbf{K}_{\mathbf{X}\mathbf{Z}_\beta}\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}\mathbf{m}_\beta, \\ \Sigma_\mathbf{ff} &= \mathbf{K}_{\mathbf{X}\mathbf{X}} + \mathbf{K}_{\mathbf{X}\mathbf{Z}_\gamma}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}(\mathbf{S}_\gamma - \mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma})\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{X}}. \end{aligned} \quad (38.18)$$

Moreover, the KL regulariser in the ELBO objective (Eq. 38.12) becomes

$$\text{KL}[q(\mathbf{u})||p(\mathbf{u}|\mathbf{Z})] = \frac{1}{2}\mathbf{m}_\beta^\top\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}\mathbf{m}_\beta + \frac{1}{2}\text{Tr}[\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma - \mathbf{I}] - \frac{1}{2}\log\det(\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma). \quad (38.19)$$

In addition, since in both the posterior predictive distribution, $q(\mathbf{f}|\mathbf{X}, \mathbf{Z})$, and the KL regulariser, the variational mean parameter, $\mathbf{m}_\beta$, always appears together with a left matrix multiplication of $\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}$, we can reparameterise the variational mean as $\mathbf{v}_\beta := \mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}\mathbf{m}_\beta$, which is then treated as a final variational parameter to optimise. As a result, the posterior predictive mean and the KL regulariser can be rewritten as

$$\begin{aligned} \mathbf{m}_\mathbf{f} &= \mathbf{K}_{\mathbf{X}\mathbf{Z}_\beta}\mathbf{v}_\beta, \\ \text{KL}[q(\mathbf{u})||p(\mathbf{u}|\mathbf{Z})] &= \frac{1}{2}\mathbf{v}_\beta^\top\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}\mathbf{v}_\beta + \frac{1}{2}\text{Tr}[\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma - \mathbf{I}] - \frac{1}{2}\log\det(\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}^{-1}\mathbf{S}_\gamma). \end{aligned} \quad (38.20)$$

Notice that with this reparameterisation, the expensive matrix inversion and determinant evaluation are now all computed for $\mathbf{K}_{\mathbf{Z}_\gamma\mathbf{Z}_\gamma}$ based on the second set of inducing points, $\{(\mathbf{Z}_\gamma, \mathbf{u}_\gamma)\}$, of size $M_\gamma := |\gamma| < M$, allowing further reduction of computation cost. On the other hand, one can consider using a larger number of inducing points $(\mathbf{Z}_\beta, \mathbf{u}_\beta)$ to ensure a more accurate posterior predictive mean, since now the time complexity of the posterior predictive distribution scales linearly w.r.t. M_β as shown in Table 38.1.

Salimbeni et al. [1454] further improved the optimisation of DSVGP by considering an alternative structured decoupling that admits an efficient natural gradient [26] update rule for faster convergence. The resulting approach, orthogonally decoupled SVGP (ODSVGP), considers the following decoupled variational distribution design:

$$\begin{aligned} q(\mathbf{u}) &= q(\mathbf{u}_\gamma)q(\mathbf{u}_\beta|\mathbf{u}_\gamma) \\ &= \mathcal{N}(\mathbf{u}_\gamma; \mathbf{m}_\gamma, \mathbf{S}_\gamma)\mathcal{N}(\mathbf{u}_\beta; \mathbf{C}\mathbf{u}_\gamma + \mathbf{B}\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}\mathbf{m}_\beta, \mathbf{B}), \end{aligned} \quad (38.21)$$

which admits an alternative decoupled parameterisation of the variational parameters

$$\mathbf{m}_\mathbf{u} = \begin{pmatrix} \mathbf{m}_\gamma \\ \mathbf{C}\mathbf{m}_\gamma + \mathbf{B}\mathbf{K}_{\mathbf{Z}_\beta\mathbf{Z}_\beta}^{-1}\mathbf{m}_\beta \end{pmatrix}, \quad \mathbf{S}_\mathbf{u} = \begin{pmatrix} \mathbf{S}_\gamma & \mathbf{S}_\gamma\mathbf{C}^\top \\ \mathbf{C}\mathbf{S}_\gamma & \mathbf{B} + \mathbf{C}\mathbf{S}_\gamma\mathbf{C}^\top \end{pmatrix}. \quad (38.22)$$

Table 38.1: Complexity comparison for GP, SVGP, DSVGP, and ODSVGP. Here N denotes the number of training points, M the total number of inducing points, and M_β , M_γ the sizes of the two complementary inducing subsets used in the decoupled variants (with $M_\beta + M_\gamma = M$).

Model	Time complexity	Memory complexity
GP [1382]	$O(N^3)$	$O(N^2)$
SVGP [737, 738]	$O(NM^2 + M^3)$	$O(M^2)$
DSVGP [285]	$O(NM_\beta + NM_\gamma^2 + M_\gamma^3)$	$O(M_\beta + M_\gamma^2)$
ODSVGP [1454]	$O(NM_\gamma M_\beta + NM_\gamma^2 + M_\gamma^3)$	$O(M_\beta + M_\gamma^2)$

The posterior predictive covariance $\Sigma_{\mathbf{ff}}$ is still the same as in Eq. 38.18, but the posterior predictive mean is now computed based on all the inducing points. With the reparameterisation $\mathbf{v}_\beta := \mathbf{K}_{\mathbf{z}_\beta \mathbf{z}_\beta}^{-1} \mathbf{m}_\beta$ and $\mathbf{v}_\gamma := \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma}^{-1} \mathbf{m}_\gamma$, the posterior predictive mean is computed as

$$\mathbf{m}_{\mathbf{f}} = \mathbf{K}_{\mathbf{XZ}_\beta} \mathbf{v}_\beta - \mathbf{K}_{\mathbf{XZ}_\gamma} \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma}^{-1} \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\beta} \mathbf{v}_\beta + \mathbf{K}_{\mathbf{XZ}_\gamma} \mathbf{v}_\gamma. \quad (38.23)$$

The KL regulariser becomes

$$\begin{aligned} \text{KL}[q(\mathbf{u})||p(\mathbf{u}|\mathbf{Z})] &= \frac{1}{2} \mathbf{v}_\beta^\top [\mathbf{K}_{\mathbf{z}_\beta \mathbf{z}_\beta} - \mathbf{K}_{\mathbf{z}_\beta \mathbf{z}_\gamma} \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma}^{-1} \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\beta}] \mathbf{v}_\beta + \frac{1}{2} \mathbf{v}_\gamma^\top \mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma} \mathbf{v}_\gamma \\ &\quad + \frac{1}{2} \text{Tr}[\mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma}^{-1} \mathbf{S}_\gamma - \mathbf{I}] - \frac{1}{2} \log \det(\mathbf{K}_{\mathbf{z}_\gamma \mathbf{z}_\gamma}^{-1} \mathbf{S}_\gamma). \end{aligned} \quad (38.24)$$

The follow-up work by Shi et al. [1513] further proposed a generalised decoupled SVGP parameterisation, which recovers ODSVGP as a special case. In Table 38.1, we summarise the time and memory complexity for full GP, SVGP, DSVGP, and ODSVGP.

38.4 Interdomain Gaussian processes

Standard SVGP defines inducing variables as the function values evaluated at some inducing locations $\mathbf{Z}$ that live in the same space as the inputs $\mathbf{X}$. Interdomain Gaussian processes [987, 1663] further consider generalised inducing variables defined through transformations of the prior GP. Each inducing variable u_m is obtained by transforming the prior GP f through a linear operator $L_m[\cdot]$:

$$u_m := L_m[f], \quad p(f) = \mathcal{GP}(0, k(\cdot, \cdot)). \quad (38.25)$$

Since the variables obtained by transforming a GP with linear operators also follow a multivariate Gaussian distribution, the prior distribution of $\mathbf{u}$ is a Gaussian as in standard SVGP, enabling the extension of SVGP to leverage interdomain inducing variables.

A particularly popular class of linear transformations is linear projection using a basis function $\phi_m(\mathbf{x})$, which defines each inducing variable as

$$u_m := \int_{\mathcal{X}} f(\mathbf{x}) \phi_m(\mathbf{x}) d\mathbf{x}, \quad p(f) = \mathcal{GP}(0, k(\cdot, \cdot)). \quad (38.26)$$

Notice that standard SVGP can be recovered as a special case in this framework by considering $\phi_m(\mathbf{x}) = \delta_{\mathbf{z}_m}(\mathbf{x})$, where $\delta_{\mathbf{z}_m}(\cdot)$ is a Dirac-delta function, and therefore $u_m = \int f(\mathbf{x}) \delta_{\mathbf{z}_m}(\mathbf{x}) d\mathbf{x} = f(\mathbf{z}_m)$.

For general basis functions, the prior cross-covariance between the function values $f(\mathbf{x})$ and the inducing variables, and the prior inducing covariance between the inducing variables themselves are computed as integrals. To maintain the consistency of notations, we denote the prior cross-covariance between $f(\mathbf{x})$ and u_m , and the prior inducing covariance between u_n and u_m as $[\mathbf{K}_{f\mathbf{x}\mathbf{u}}]_m$ and $[\mathbf{K}_{\mathbf{u}\mathbf{u}}]_{nm}$, respectively:

$$\begin{aligned} [\mathbf{K}_{f\mathbf{x}\mathbf{u}}]_m &:= \text{COV}(f(\mathbf{x}), u_m) = \int_{\mathcal{X}} k(\mathbf{x}, \mathbf{x}') \phi_m(\mathbf{x}') d\mathbf{x}', \\ [\mathbf{K}_{\mathbf{u}\mathbf{u}}]_{nm} &:= \text{COV}(u_n, u_m) = \int_{\mathcal{X}} \int_{\mathcal{X}} k(\mathbf{x}, \mathbf{x}') \phi_n(\mathbf{x}) \phi_m(\mathbf{x}') d\mathbf{x} d\mathbf{x}'. \end{aligned} \quad (38.27)$$

Typically, the integration domain $\mathcal{X}$ is bounded to avoid exploding prior variances [740, 987].

Since we still impose the same structure in the approximate posterior as in standard SVGP (i.e., freely optimised variational distribution for the inducing variables and prior conditional matching), the form of posterior predictive distribution and the ELBO training objective for interdomain GPs remain exactly the same as the standard SVGP shown in Section 38.2. The only difference is that we replace the covariance matrices $\mathbf{K}_{\mathbf{XZ}}$ and $\mathbf{K}_{\mathbf{ZZ}}$ in standard SVGP with $\mathbf{K}_{\mathbf{fu}}$ and $\mathbf{K}_{\mathbf{uu}}$, respectively. The main technical contribution of most interdomain GP methods [476, 740, 987] is on finding basis functions that modulate the behaviour of GP approximations, as well as tractable computation of the integration-based covariance matrices $\mathbf{K}_{\mathbf{fu}}$ and $\mathbf{K}_{\mathbf{uu}}$.

Compared with standard SVGP which requires (iterative) optimisation of the inducing inputs $\mathbf{Z}$, interdomain GP relies on fixed basis functions for $\mathbf{K}_{\mathbf{fu}}$ and $\mathbf{K}_{\mathbf{uu}}$ computations. Therefore, interdomain GPs often incur less computational cost than standard SVGP, especially when the integrals are analytically tractable or can be computed efficiently [278]. Furthermore, with some combinations of kernels and basis functions, the prior inducing covariance $\mathbf{K}_{\mathbf{uu}}$ may enjoy certain sparse structures to enable fast matrix inversion [223, 476, 740]. Notably, Dutordoir et al. [476] constructed a diagonal $\mathbf{K}_{\mathbf{uu}}$ based on zonal kernels and spherical harmonics basis functions, so that its inversion can be computed in only $O(M)$ time.

38.5 Online sparse Gaussian processes

In these online or continual learning [250, 897, 1236, 1401] scenarios, we would like to update the model only based on the new batch of data (or the new task) without revisiting the past data, since storing all the past data and retraining the model on them become more and more expensive as more and more tasks are observed. However, the model typically suffers from catastrophic forgetting and cannot maintain consistent performance over the past data if we merely adapt it to the new task. Online or continual learning methods [347, 458, 1236] are proposed to mitigate the catastrophic forgetting issue so that the model can both adapt to the new task and also memorise the past tasks, without expensive retraining on (all the) past data.

Sequential Bayesian inference provides a principled framework for online learning where the old posterior is used as the prior knowledge for the new task. By adapting it to the new data through incorporating the corresponding new likelihood, we can then obtain the new posterior representing our updated knowledge. Approximation techniques, such as sequential variational inference (reviewed in Chapter 18), are required when exact sequential posterior inference is intractable.

For online learning with SVGPs, we sequentially update the inducing points and its variational parameters as new data arrive in the framework of sequential variational inference. Given the SVGP trained on the first task $\mathcal{D}_{t_1}$, $q_{t_1}(f) = \int p_{t_1}(f|\mathbf{u}_{t_1})q_{t_1}(\mathbf{u}_{t_1})d\mathbf{u}_{t_1}$ based on $(\mathbf{Z}_{t_1}, \mathbf{u}_{t_1})$, when the second task is observed, online SVGP [OSVGP; 218] obtain the new SVGP, $q_{t_2}(f) = \int p_{t_2}(f|\mathbf{u}_{t_2})q_{t_2}(\mathbf{u}_{t_2})d\mathbf{u}_{t_2}$, by updating the inducing points to $(\mathbf{Z}_{t_2}, \mathbf{u}_{t_2})$ as well as fitting the approximate posterior $q_{t_2}(\mathbf{u}_{t_2})$. Notice that the prior for the new SVGP can be updated as well (i.e., the kernel hyperparameters can be updated), so $p_{t_2}(f)$ can be different from $p_{t_1}(f)$. Ideally, in the case where revisiting the past data is allowed, we would like to minimise the full-batch KL-divergence $\text{KL}(q_{t_2}(f)||p_{t_2}(f|\mathcal{D}_{t_1}, \mathcal{D}_{t_2}))$, which is equivalent to maximise the following ELBO:

$$\begin{aligned}\mathcal{L}_{ELBO} &= \log \frac{p_{t_2}(\mathcal{D}_{t_1}, \mathcal{D}_{t_2})}{p_{t_1}(\mathcal{D}_{t_1})} - \text{KL}(q_{t_2}(f)||p_{t_2}(f|\mathcal{D}_{t_1}, \mathcal{D}_{t_2})) \\ &= \int q_{t_2}(f) \log \frac{p_{t_2}(\mathbf{y}_{t_2}, \mathbf{y}_{t_1})}{p_{t_1}(\mathbf{y}_{t_1})q_{t_2}(f)} p_{t_2}(f|\mathbf{y}_{t_1}, \mathbf{y}_{t_2}) df \\ &= \int q_{t_2}(f) \log \frac{p_{t_2}(f)p_{t_2}(\mathbf{y}_{t_2}|f)p_{t_2}(\mathbf{y}_{t_1}|f)}{p_{t_1}(\mathbf{y}_{t_1})q_{t_2}(f)} df.\end{aligned}\tag{38.28}$$

Notice that $p_{t_1}(\mathcal{D}_{t_1})$ in the denominator of the first term is a constant since it only depends on the old kernel hyperparameters. Unfortunately, in an online setup, since we are not allowed to revisit the old data with the new model, $p_{t_2}(\mathbf{y}_{t_1}|f)$ is intractable and therefore the above ELBO is intractable. To get around this intractability, Bui et al. [218] proposed to approximate this term with

$$p_{t_2}(\mathbf{y}_{t_1}|f) \approx p_{t_1}(\mathbf{y}_{t_1}|f) = \frac{p_{t_1}(f|\mathbf{y}_{t_1})p_{t_1}(\mathbf{y}_{t_1})}{p_{t_1}(f)} \approx \frac{q_{t_1}(f)p_{t_1}(\mathbf{y}_{t_1})}{p_{t_1}(f)}.\tag{38.29}$$

Based on this approximation, the above ELBO in Eq. 38.28 can be approximated with the following “online ELBO”:

$$\mathcal{L}_{ELBO}^o = \int q_{t_2}(f) \log \frac{p_{t_2}(f)p_{t_2}(\mathbf{y}_{t_2}|f)q_{t_1}(f)}{q_{t_2}(f)p_{t_1}(f)} df.\tag{38.30}$$

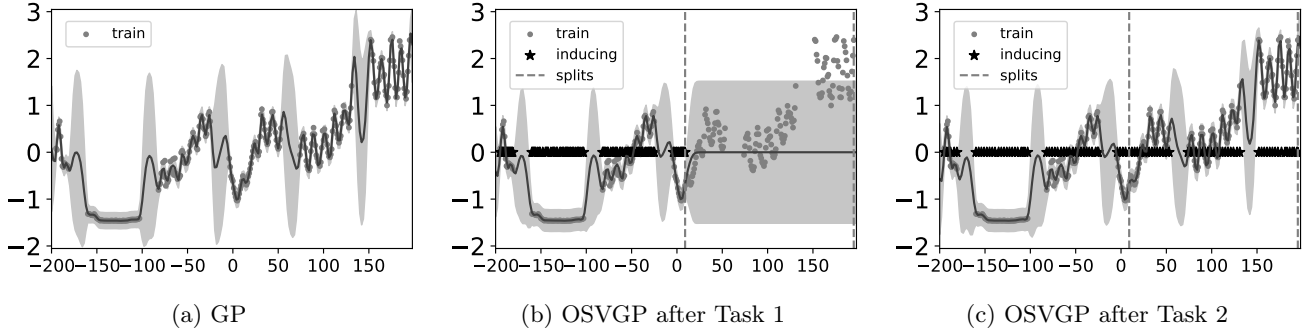


Figure 38.2: OSVGP with 100 inducing points on an online time series regression problem consisting of two tasks. The dash lines in panels (b) and (c) indicate the split of data into two tasks.

Since $q_{t_1}(f)$ and $q_{t_2}(f)$ are SVGPs fully specified by inducing points $\mathbf{u}_{t_1}$ and $\mathbf{u}_{t_2}$, respectively, the objective can be further simplified to:

$$\begin{aligned} \mathcal{L}_{ELBO}^o = & \mathbb{E}_{q_{t_2}(\mathbf{f}_{t_2})} [\log p_{t_2}(\mathbf{y}_{t_2} | \mathbf{f}_{t_2})] + \text{KL}(\tilde{q}_{t_2}(\mathbf{u}_{t_1}) \| p_{t_1}(\mathbf{u}_{t_1})) \\ & - \text{KL}(\tilde{q}_{t_2}(\mathbf{u}_{t_1}) \| q_{t_1}(\mathbf{u}_{t_1})) - \text{KL}(q_{t_2}(\mathbf{u}_{t_2}) \| p_{t_2}(\mathbf{u}_{t_2})), \end{aligned} \quad (38.31)$$

where $\tilde{q}_{t_2}(\mathbf{u}_{t_1}) := \int p_{t_2}(\mathbf{u}_{t_1} | \mathbf{u}_{t_2}) q_{t_2}(\mathbf{u}_{t_2}) d\mathbf{u}_{t_2}$.

The time complexity of online SVGP is $O(BM^2 + M^3)$ per step, where B is the size of the streaming data batch (analogous to the batch size N for SVGP). Figure 38.2 shows the fit of OSVGP for online regression on the Solar Irradiance dataset [992]. The time series is split into two tasks and OSVGP learns these two tasks online. When learning the second task, even without retraining on the first task, OSVGP still maintains reliable predictive performance in this historical region and also adapts to the new task. The final prediction is similar to a full GP model trained jointly over all the data, illustrating OSVGP's efficiency.

Other approximate inference techniques, such as Laplace approximation [1118], expectation propagation [218, 347], and variational approximation enhanced with replay buffer, have also been proposed to bypass the intractability of exact online posterior inference [257].

Interdomain GPs can also be integrated into the online SVGP framework [278] and since the interdomain inducing variables are constructed without training the inducing locations, they can further reduce the computational cost. However, previous interdomain GPs typically construct inducing variables via integration based on a predefined measure (e.g., a uniform measure over a fixed interval [740] or a fixed Gaussian measure [987]) to prevent exploding prior variances. This requires the practitioner to pre-define an integration measure that covers all regions where the future tasks live, which is impractical for real-world online learning problems. Chen et al. [278] proposed online HiPPO-SVGP (OHSVGP), which bypasses this limitation by adopting adaptive basis functions in the HiPPO (High-order Polynomial Projection Operators) framework [678] (a renovated RNN architecture). These adaptive basis functions are constructed based on a time-dependent measure which keeps extending to the new input region as more tasks arrive, making them suitable for online inference. As a result, the prior covariance matrices, $\mathbf{K}_{\text{fu}}$ and $\mathbf{K}_{\text{uu}}$, are now adaptive as well, and they can be efficiently updated in a recurrence form when a new task arrives, similar to the evolution of the recurrent hidden state in the HiPPO framework.

38.6 Summary

This chapter reviews scalable GP methods. We have discussed scalable low-rank approximations of kernel matrix enabling faster computation of matrix inversion and determinant evaluation in full GP. Then we reviewed SVGPs based on variational inducing points, arguably the most representative technique for scaling up GP posterior inference. We also introduced decoupled SVGPs and interdomain GPs as extensions of standard SVGPs for better scalability and approximation quality. Lastly, we discussed online SVGP methods that enable scalable online learning with GP models, which further reduce the computational cost by avoiding retraining over historical data. This chapter is by no means exhaustive, but most methods included here are applicable to general setups, e.g., regression, classification, and latent variable models (e.g., Titsias and Lawrence [GP-LVM; 1616]). While we have focused on variational approaches to sparse GP inference, MCMC-based alternatives also exist. These methods place priors over the inducing variables or kernel hyperparameters, and sample from their posterior, and can offer improved uncertainty quantification at additional computational cost [739, 963, 1425, 1647]. Further improvements

of scalable GPs based on task-specific requirements are possible. For example, Markovian GPs [1460, Chapter 12] exploit their state-space SDE representation to reduce the time complexity to $\mathcal{O}(N)$, but these approaches are restricted to 1D input and Markovian kernels.

Chapter 39

Last-layer inference

In the previous chapter, we saw that Gaussian processes scale cubically with the number of data points, and one way to lower this cost is to reduce the effective number of data points via inducing points. An analogous strategy can be applied to Bayesian neural networks, where the computational bottleneck lies in the number of parameters. Instead of computing a posterior over all network weights, one can compute a posterior over the last-layer weights only and treat the remaining parameters as deterministic. This approach is called *last-layer inference* and is the topic of this chapter.

39.1 Motivation

Many drawbacks of deterministic neural networks can be addressed by treating all network weights as random variables and performing posterior estimation using Bayesian inference. In classification, Bayesian neural networks mitigate overconfidence and improve calibration of predictive class probabilities, whereas in regression they provide principled predictive densities that are not available for deterministic neural networks. Yet, such fully Bayesian neural networks are rarely used in practice. This is because they do not scale well to the large datasets and highly parameterised neural networks required for real-world applications. In the simplest fully Bayesian neural network, one not only has to learn the weights but also the corresponding posterior variances. Thus, one has to learn twice as many parameters as in a deterministic neural network. Ad-hoc approximate methods, such as Monte Carlo (MC) dropout, are a practical solution to make computation feasible, but require several forward passes during prediction and make it difficult to encode prior assumptions in a principled manner.

In some applications, practitioners are willing to trade off accurate inference (in the form of fully Bayesian neural networks) against computational speed in training and prediction. To that end, it is often sufficient to only perform Bayesian inference over a *subset* of layers or weights. A popular choice in that regard is to infer the posterior over the last-layer weights $w \in \mathbb{R}^{C \cdot H_L - 1}$ and use point estimates for the remaining weights $\theta \in \mathbb{R}^{P - C \cdot H_L - 1}$. This is called *last-layer inference*.

Assume we wish to predict a target $y_* \in \mathbb{R}^C$ from an input $x_* \in \mathbb{R}^D$ using a neural network $f_\vartheta : \mathbb{R}^D \rightarrow \mathbb{R}^C$ with parameters $\vartheta = (\theta^\top, w^\top)^\top \in \mathbb{R}^P$ trained on observed data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$.

In last-layer inference, one approximates the posterior predictive distribution via

$$p(y_* | x_*, \mathcal{D}) \approx p(y_* | x_*, \mathcal{D}, \theta) = \int p(y_* | x_*, \mathcal{D}, \theta, w) p(w | \mathcal{D}, \theta) dw. \quad (39.1)$$

This approach offers two primary benefits. Computationally, it reduces the dimensionality of the posterior inference task, which speeds up training and prediction compared to a fully Bayesian neural network. Statistically, it can be interpreted as performing Bayesian linear regression with learned deep feature projections as covariates. This is a setting where closed-form posterior solutions with conjugate priors exist, and efficient algorithms for various other prior choices are well established. However, it is important to note that the quality of uncertainty estimates produced by last-layer inference may, in some applications, depend on the representational capacity of the deep feature projections, and ignoring non-determinism from earlier layers may lead to different uncertainty estimates. We illustrate the main concept of last-layer inference in Fig. 39.1.

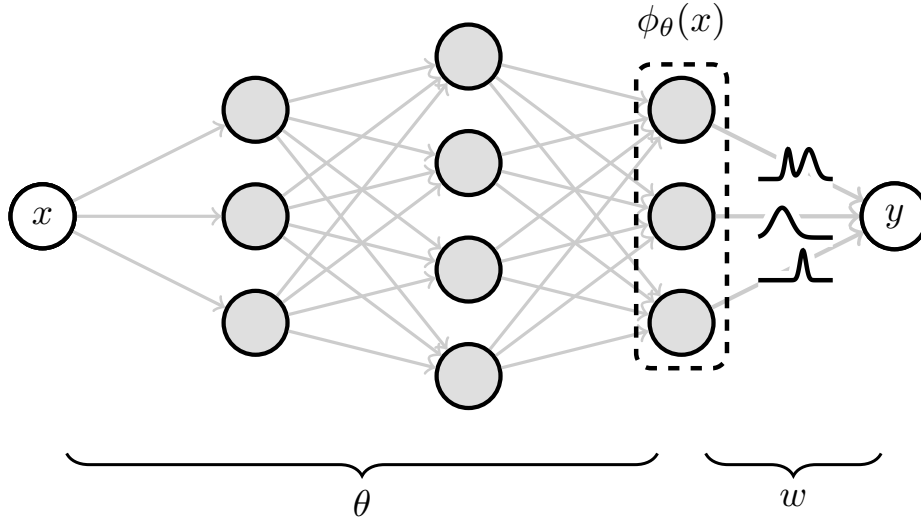


Figure 39.1: In last-layer inference, the parameters $\theta \in \mathbb{R}^{P-C \cdot H_{L-1}}$ up to and including the penultimate layer are treated as deterministic. Only the posterior $p(w \mid \mathcal{D})$ for the last-layer weights is computed using the deep feature projection ϕ_θ . Figure inspired by Blundell et al. [174].

39.2 Last-layer inference as a Bayesian linear model

Last-layer inference can be seen as a Bayesian linear regression model on deep feature projections of the inputs x . For simplicity, we focus on regression for scalar targets, such that $C = 1$ and $w \in \mathbb{R}^{H_{L-1}}$. To perform last-layer inference for classification, we refer to Chapter 10, which treats Laplace approximations for classification, to Gelman et al. [601] for an overview of Bayesian models for classification, as well as to Harrison et al. [709] for an example of last-layer inference based on a Dirichlet distribution.

Let $\phi_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{H_{L-1}}$ denote the sub-network of f up to the penultimate layer, which maps an input x to a deep feature projection $\phi_\theta(x)$. The network ϕ_θ is called a deep feature projector because it maps the input features to a lower-dimensional space, since often $H_{L-1} \ll D$. As shorthand notation, the deep feature projection at a specific training input x_n for $n = 1, \dots, N$ is denoted as $\phi_n := \phi_\theta(x_n)$.

The output of the neural network using a fully connected last layer with a linear activation can be formulated as a linear regression model with covariates ϕ_θ . For an arbitrary input x the neural network output can be written as

$$f_\theta(x) := w^\top \phi_\theta(x), \quad (39.2)$$

where a constant is appended to $\phi_\theta(x)$ to account for the bias term.

While θ are assumed to be deterministic, w are interpreted as random variables with prior distribution $p(w)$. We are interested in inferring the respective posterior distribution $p(w \mid \mathcal{D}, \theta)$ after observing training data $\mathcal{D}$. The latter can be obtained by applying Bayes' rule

$$p(w \mid \mathcal{D}, \theta) = \frac{p(\mathcal{D} \mid \theta, w)p(w)}{p(\mathcal{D} \mid \theta)}, \quad (39.3)$$

where $p(\mathcal{D} \mid \theta, w)$ defines the likelihood and $p(\mathcal{D} \mid \theta)$ is the marginal likelihood or evidence. In Eq. (39.1), we assume that θ are deterministic when computing the posterior predictive distribution, that is, $p(w \mid \mathcal{D}) = p(w \mid \mathcal{D}, \theta)$. Therefore, we drop θ in the following and write the posterior as $p(w \mid \mathcal{D})$ and the evidence as $p(\mathcal{D})$ instead.

To perform Bayesian inference in the model (39.2), it is necessary to specify a likelihood, a prior on w , and to apply Bayes' rule to obtain the posterior distribution. We first discuss how to extend Eq. (39.2) to obtain a likelihood, give an overview of suitable priors w , and derive the resulting posterior distributions.

39.2.1 Likelihood

The likelihood in the canonical Bayesian linear regression model is a noisy version of expression (39.2), that is,

$$y_n = w^\top \phi_n + \varepsilon_n, \quad (39.4)$$

where $\varepsilon_n \sim \mathcal{N}_1(0, \sigma_\varepsilon^2)$ are i.i.d. Gaussian errors, and $\mathcal{N}_1(m, s)$ denotes a one-dimensional Gaussian distribution with mean m and variance s . More complex noise distributions are possible (see Sec. 39.5.1), but for simplicity only the homoscedastic case is considered throughout this chapter. In matrix form, the likelihood over the entire training set is

$$p(\mathcal{D} \mid w, \sigma_\varepsilon^2) \equiv p(Y \mid X, w, \sigma_\varepsilon^2) = \mathcal{N}(Y; \Phi w, \sigma_\varepsilon^2 I_N) = \prod_{n=1}^N \mathcal{N}(y_n; w^\top \phi_n, \sigma_\varepsilon^2), \quad (39.5)$$

where $\mathcal{N}(x; m, S)$ denotes the density of a Gaussian distribution with mean m and covariance S , the matrix I_N is the N -dimensional identity matrix, $\Phi = [\phi_1^\top, \dots, \phi_N^\top] \in \mathbb{R}^{N \times H_{L-1}}$ are the vertically stacked row vectors of the deep feature projections at the training inputs $X = [x_1^\top, \dots, x_N^\top]^\top \in \mathbb{R}^{D \times N}$, the vector $Y = (y_1, \dots, y_N)^\top \in \mathbb{R}^N$ contains the training targets. To perform Bayesian inference in the model (39.4), we have to choose a prior for the last-layer weights w and the variance of the observational noise σ_ε^2 .

39.2.2 Prior distributions

We focus here on the most common priors for w , how to use them to induce shrinkage and regularisation, and we present a default prior for σ_ε^2 . Further details on priors in general can be found in Part VI (see in particular Chapters 28 and 29).

Prior for the last-layer weights w

We assume a Gaussian prior for the last-layer weights, that is,

$$w \sim \mathcal{N}(\mu_0, \Sigma_0) \quad (39.6)$$

with mean $\mu_0 \in \mathbb{R}^{H_{L-1}}$ and covariance $\Sigma_0 \in \mathbb{R}^{H_{L-1} \times H_{L-1}}$. The Gaussian prior is conjugate to the likelihood in Eq. (39.5), i.e., the posterior is of the same distributional family as the prior distribution, which is attractive from a computational point of view. In the absence of problem-specific prior knowledge, the mean μ_0 is commonly set to zero. The covariance can be assigned a hyperprior that induces shrinkage to reduce overfitting and separate signal from noise, and we now introduce two such designs.

The most common shrinkage prior for Σ_0 is a ridge prior of the form

$$\Sigma_0 = \tau^2 I, \quad \text{where } \tau^2 \sim \mathcal{IG}(a_\tau, b_\tau), \quad (39.7)$$

where $\tau^2 > 0$ denotes a scalar variance parameter, $\mathcal{IG}$ denotes the inverse gamma distribution with shape $a_\tau > 0$ and scale $b_\tau > 0$. Appropriate choices for the parameters a_τ and b_τ of the hyperprior can be found in Gelman [596] and George and McCulloch [607].

The inverse gamma prior for τ^2 is conjugate, and encompasses weakly informative priors on the log-scale ($a_\tau = \epsilon, b_\tau = \epsilon, \epsilon \rightarrow 0$), as well as flat priors for the variance ($a_\tau = -1, b_\tau = 0$) and standard deviation ($a_\tau = -1/2, b_\tau = 0$) as degenerate cases.

However, there has been considerable debate about the suitability of the inverse gamma distribution due to overfitting [1516]. Thus, alternative choices such as half-Normal, half-Cauchy [1336] or scale-dependent priors [901] have been proposed. Fig. 39.2 depicts the densities of the different hyperpriors for the variance parameter.

A further popular prior choice that induces more finely controlled sparsity is the horseshoe prior [243]. The horseshoe prior introduces independent Gaussian priors for each last-layer weight

$$w_l \sim \mathcal{N}(0, \tau_l^2), \quad \text{for } l = 1, \dots, H_{L-1} \quad (39.8)$$

and a global-local shrinkage structure

$$\tau_l \mid \xi \sim \mathcal{HC}(0, \xi), \quad \text{with } \xi \sim \mathcal{HC}(0, b_\xi), \quad (39.9)$$

where $\mathcal{HC}$ denotes the half-Cauchy distribution and $b_\xi > 0$ is a fixed hyperparameter. Consequently, the covariance matrix is diagonal $\Sigma_0 = \text{diag}(\tau_1^2, \dots, \tau_{H_{L-1}}^2)$, and the diagonal elements τ_l^2 allow for varying amounts of regularisation across the weight dimension. Choosing half-Cauchy distributions results in strong shrinkage, such that this choice is appropriate when, effectively, most of the weights should be shrunk towards zero [1123]. The heavy tails of the half-Cauchy distribution allow sufficiently large weights to remain un-shrunk [243]. This can, for instance, be helpful when the last layer is heavily overparameterised.

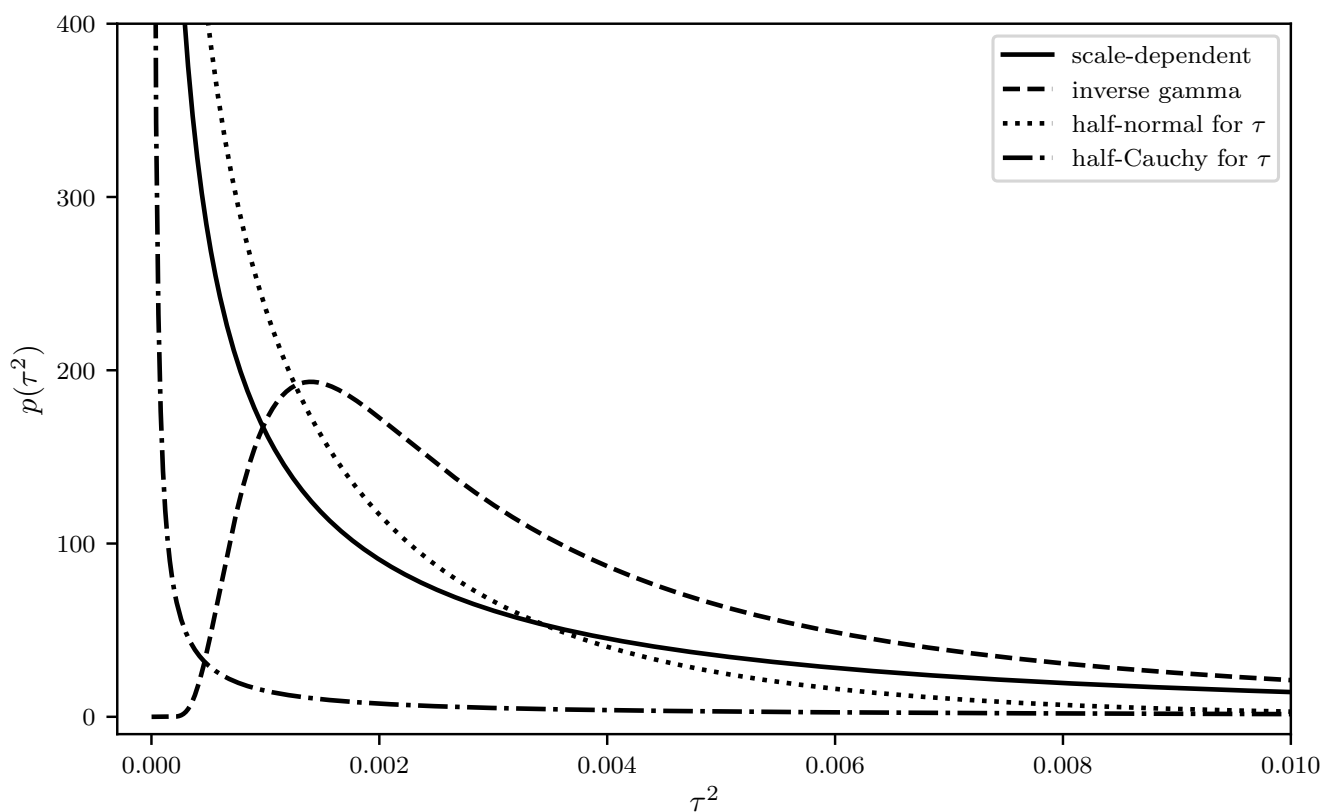


Figure 39.2: Densities for the hyperpriors on the variance parameter τ^2 . The half-Cauchy prior on τ corresponds to a generalised beta prime prior on τ^2 , and the half-normal prior for τ corresponds to a gamma prior for τ^2 . The figure is adapted from Klein and Kneib [901].

Prior for the error noise σ_ε^2

In the standard Bayesian linear regression model, the variance of the error noise is assumed to have an inverse gamma prior

$$\sigma_\varepsilon^2 \sim \mathcal{IG}(a_\sigma, b_\sigma), \quad (39.10)$$

with $a_\sigma, b_\sigma > 0$ being the shape and scale parameters. Similar to the prior for the weights, this choice is conjugate and thus yields a closed-form (conditional) posterior distribution. It also ensures positivity of the variance due to the positive support of the inverse gamma distribution.

39.2.3 Posterior distributions

By Bayes' rule, the joint posterior $p(w, \sigma_\varepsilon^2, \Sigma_0 \mid \mathcal{D})$ is proportional to

$$p(w, \sigma_\varepsilon^2, \Sigma_0 \mid \mathcal{D}) \propto p(\mathcal{D} \mid w, \sigma_\varepsilon^2) p(w \mid \Sigma_0) p(\Sigma_0) p(\sigma_\varepsilon^2). \quad (39.11)$$

The $\propto$ symbol indicates proportionality up to factors not depending on the model parameters $w, \sigma_\varepsilon^2, \Sigma_0$. A key observation is that Bayesian inference usually builds upon the last equation such that the intractable evidence $p(\mathcal{D})$ is neglected in further computations. We now derive the conditional posteriors, which are, for example, used for the Gibbs sampler in Sec. 39.3.

Conditional posterior of w

The conditional posterior of the last-layer weights is

$$p(w \mid \mathcal{D}, \sigma_\varepsilon^2, \Sigma_0) \propto p(\mathcal{D} \mid w, \sigma_\varepsilon^2) p(w \mid \Sigma_0) \quad (39.12)$$

$$= \mathcal{N}(Y; \Phi w, \sigma_\varepsilon^2 I) \mathcal{N}(w; \mu_0, \Sigma_0) \quad (39.13)$$

$$= \mathcal{N}(w; \mu_N, \Sigma_N) \quad (39.14)$$

with posterior mean μ_N and covariance Σ_N given by

$$\mu_N = \Sigma_N (\Sigma_0^{-1} \mu_0 + \sigma_\varepsilon^{-2} \Phi^\top Y) \text{ and } \Sigma_N = (\Sigma_0^{-1} + \sigma_\varepsilon^{-2} \Phi^\top \Phi)^{-1}. \quad (39.15)$$

Conditional posterior of σ_ε^2

The conditional posterior of the observational noise is

$$p(\sigma_\varepsilon^2 \mid \mathcal{D}, w) \propto p(\mathcal{D} \mid w, \sigma_\varepsilon^2) p(\sigma_\varepsilon^2). \quad (39.16)$$

With the conditionally conjugate inverse gamma prior from Eq. (39.10), the conditional posterior is also inverse gamma distributed

$$p(\sigma_\varepsilon^2 \mid \mathcal{D}, w) \propto \mathcal{N}(Y; \Phi w, \sigma_\varepsilon^2 I) \cdot \mathcal{IG}(\sigma_\varepsilon^2; a_\sigma, b_\sigma) \quad (39.17)$$

$$\propto \mathcal{IG}(\sigma_\varepsilon^2; a_\sigma + \frac{N}{2}, b_\sigma + \frac{1}{2} \|Y - \Phi w\|^2), \quad (39.18)$$

where $\mathcal{IG}(x; a, b)$ denotes the density of the inverse gamma distribution with shape a and scale b evaluated at x .

Conditional posterior of Σ_0

The conditional posterior for the covariance Σ_0 of the last-layer weight distribution is

$$p(\Sigma_0 \mid \mathcal{D}, w, \sigma_\varepsilon^2) \propto p(w \mid \Sigma_0) p(\Sigma_0), \quad (39.19)$$

using the conditional independence assumption $\Sigma_0 \perp (\mathcal{D}, \sigma_\varepsilon^2) \mid w$. With the ridge prior and setting $\mu_0 = 0$, this conditional posterior is the posterior of τ^2 , which is also inverse gamma distributed

$$p(\tau^2 \mid \mathcal{D}, w, \sigma_\varepsilon^2) \propto \mathcal{N}(w; 0, \tau^2 I_{H_{L-1}}) \mathcal{IG}(\tau^2; a_\tau, b_\tau) \quad (39.20)$$

$$\propto \mathcal{IG}(\tau^2; a_\tau + \frac{H_{L-1}}{2}, b_\tau + \frac{1}{2} \|w\|^2). \quad (39.21)$$

Let $\mathcal{HC}(x; 0, b)$ denote the density of the half-Cauchy distribution with location 0 and scale b evaluated at x . Then, with the horseshoe prior the conditional posterior of $\tau_1^2, \dots, \tau_{H_{L-1}}^2, \xi$ is

$$p(\tau_1^2, \dots, \tau_{H_{L-1}}^2, \xi \mid \mathcal{D}, w, \sigma_\varepsilon^2) \propto \prod_{l=1}^{H_{L-1}} \mathcal{N}(w_l; 0, \tau_l^2) \mathcal{HC}(\tau_l; 0, \xi) \mathcal{HC}(\xi; 0, b_\xi), \quad (39.22)$$

which does not correspond to a closed-form distribution.

Note that, except for the horseshoe prior, we can express all conditional posteriors in closed form. Thus, we can sample from the conditional posteriors, and we will see in the next section that this allows us to efficiently obtain exact samples of the posterior using a Gibbs sampler.

39.3 Posterior estimation and prediction

To estimate the posterior distribution $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$, we briefly review Markov chain Monte Carlo (MCMC) simulation as an exact sampling method and variational inference (VI) as an approximate, optimisation-based inference method that better scales to large datasets and highly parameterised models. A third alternative, which is described in detail in Chapter 40, is the Laplace approximation, which uses a local Gaussian approximation. We then describe how to conduct two-step and joint estimation of the posterior $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$ and deterministic neural network weights θ , and how to perform predictive inference for new inputs x_* .

39.3.1 Markov chain Monte Carlo

MCMC produces asymptotically exact samples for posteriors that are known up to a normalising constant by constructing a Markov chain whose stationary distribution is the posterior. The chain is initialised and run until convergence. Usually, draws from the pre-convergence phase (burn-in period) are discarded, and the remaining draws are used as exact samples from the posterior distribution. In principle, any MCMC method from Part I can be used for last-layer inference.

We can distinguish between standard MCMC methods and those that use stochastic gradients (SG-MCMC). The reason for separating these classes is that, with SG-MCMC, we can perform *full training*, that is, joint estimation of $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$ and θ (compare Sec. 39.3.3), whereas standard MCMC only allows to fit the posterior after the network parameters θ have been learned.

Standard MCMC

To estimate $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$, the training data is first passed through the deep feature projector. Then, an MCMC sampler is run with the feature projections as covariates. Appropriate samplers are chosen based on the specified model and the dimension of the last layer. For example, in the simple, conjugate case with a Gaussian prior on the weights $w \sim \mathcal{N}(0, \tau^2 I_{H_{L-1}})$, an inverse gamma prior on the error noise $\sigma_\varepsilon^2 \sim \mathcal{IG}(a_\sigma, b_\sigma)$, and an inverse gamma hyperprior on the variance parameter $\tau^2 \sim \mathcal{IG}(a_\tau, b_\tau)$, a Gibbs sampler can be used to sample from the joint posterior $p(w, \tau^2, \sigma_\varepsilon^2 \mid \mathcal{D})$. This sampler is fast and computationally efficient and summarised in Alg. 31.

Algorithm 31: Gibbs sampler

Input : Training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$
Output: Posterior samples of $w, \tau^2, \sigma_\varepsilon^2$
for $s = 1, \dots, S$ **do**
 Draw $w \mid \mathcal{D}, \tau^2, \sigma_\varepsilon^2 \sim \mathcal{N}(\mu_N, \Sigma_N)$
 Draw $\tau^2 \mid \mathcal{D}, w, \sigma_\varepsilon^2 \sim \mathcal{IG}(a_\tau + H_{L-1}/2, b_\tau + \frac{1}{2}\|w\|^2)$
 Draw $\sigma_\varepsilon^2 \mid \mathcal{D}, \tau^2, w \sim \mathcal{IG}(a_\sigma + N/2, b_\sigma + \frac{1}{2}\|Y - \Phi w\|^2)$
return $\{w^{[s]}, (\tau^2)^{[s]}, (\sigma_\varepsilon^2)^{[s]}\}_{s=S_{\text{burn-in}}+1}^S$

A general drawback of standard MCMC methods for last-layer inference is that they use the full training set in each iteration, which can become expensive for large datasets. An alternative is to employ mini-batches, which can, for example, be combined with Hamiltonian Monte Carlo (HMC) methods [315].

SG-MCMC

SG-MCMC naturally accommodates mini-batch training, making it well-suited to large datasets. It is attractive for last-layer inference as it is conceptually simple and requires only a small, computationally inexpensive modification to the parameter updates used to train deterministic neural networks. The general idea of SG-MCMC is to add noise to stochastic gradient iterates and gradually decrease the step size during training towards zero, a process known as *annealing*. Asymptotically, the iterates will converge to samples of the posterior. A general approach for converting standard MCMC samplers into SG-MCMC is shown in Ma et al. [1097], and Chapters 2 and 5 provide further details.

A popular example of SG-MCMC are the stochastic gradient Langevin dynamics [SGLD; 1737], which updates the last-layer weights according to

$$w^{[t+1]} = w^{[t]} + \frac{\eta^{[t]}}{2} \left(\nabla_w \log p(w^{[t]}) + \frac{N}{|B|} \sum_{n \in B} \nabla_w \log p(y_n | w^{[t]}, x_n) \right) + \nu^{[t]}, \quad (39.23)$$

where t denotes the iteration, $\eta^{[t]}$ is a sequence of learning rates or so-called *step sizes*, $\nu^{[t]} \sim \mathcal{N}(0, \eta^{[t]} I_{H_{L-1}})$ is injected noise, and $B \subset \{1, \dots, N\}$ denotes the indices of a mini-batch.

The key idea of SGLD is that for relatively large values of $\eta^{[t]}$, Eq. (39.23) performs standard stochastic gradient updates, but as $\eta^{[t]} \rightarrow 0$, the updates approach Langevin dynamics and yield samples of the posterior distribution. The sequence of step sizes typically needs to satisfy $\sum_{t=1}^{\infty} \eta^{[t]} = \infty$ and $\sum_{t=1}^{\infty} (\eta^{[t]})^2 < \infty$. The first condition allows high-probability regions to be reached regardless of the initialisation, while the second controls the rate at which the injected noise vanishes.

39.3.2 Variational inference

In VI, we approximate the intractable posterior $p(w, \Sigma_0, \sigma_\epsilon^2 | \mathcal{D})$ with a tractable variational distribution with density $q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)$ that is parameterised by so-called variational parameters $\lambda \in \mathbb{R}^K$. We seek to find the parameter values λ^* that minimise a divergence measure between the posterior $p(w, \Sigma_0, \sigma_\epsilon^2 | \mathcal{D})$ and the parametric variational density $q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)$. Part III provides an in-depth introduction to VI, and we briefly derive an appropriate VI objective for last-layer inference here.

Commonly, VI is based on the reverse Kullback-Leibler (KL) divergence (compare Chapter 8), which leads to the following optimisation problem

$$\lambda^* = \arg \min_{\lambda \in \mathbb{R}^K} D_{\text{KL}}(q_\lambda(w, \Sigma_0, \sigma_\epsilon^2) \| p(w, \Sigma_0, \sigma_\epsilon^2 | \mathcal{D})). \quad (39.24)$$

Rather than minimising the KL divergence, which is typically intractable, one can equivalently maximise the evidence lower bound (ELBO), given by

$$\begin{aligned} \text{ELBO}(\lambda) &= \mathbb{E}_{q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)} [\log p(\mathcal{D} | w, \Sigma_0, \sigma_\epsilon^2)] + \mathbb{E}_{q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)} [\log p(w, \Sigma_0, \sigma_\epsilon^2)] \\ &\quad - \mathbb{E}_{q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)} [\log q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)] \\ &= \mathbb{E}_{q_\lambda(w, \Sigma_0, \sigma_\epsilon^2)} [\log p(\mathcal{D} | w, \Sigma_0, \sigma_\epsilon^2)] - D_{\text{KL}}(q_\lambda(w, \Sigma_0, \sigma_\epsilon^2) \| p(w, \Sigma_0, \sigma_\epsilon^2)) \end{aligned} \quad (39.25)$$

The ELBO can be maximised using mini-batches and stochastic gradient ascent (SGA), where the variational parameters λ at iteration $t + 1$ are updated via

$$\lambda^{[t+1]} = \lambda^{[t]} + \eta^{[t]} \widehat{\nabla_\lambda \text{ELBO}(\lambda^{[t]})}, \quad (39.26)$$

where $\widehat{\nabla_\lambda \text{ELBO}(\lambda^{[t]})}$ denotes a Monte Carlo estimate of the gradient w.r.t. the variational parameters, and $\eta^{[t]}$ is the learning rate, which may be a scalar or vector and potentially adaptive as well.

Choosing the variational distribution

Gaussian variational distributions have become popular and are known to provide generally accurate approximations of posterior means [875]. The simplest choice is the Gaussian distribution with a diagonal covariance matrix [851]. The variance parameters need to be appropriately transformed to ensure positive support. We apply a logarithmic transformation so that the elements of Σ_0 and σ_ϵ^2 follow a log-normal distribution. The resulting

variational density is

$$q_\lambda(w, \log(\Sigma_0), \log(\sigma_\varepsilon^2)) = \mathcal{N} \left(\begin{bmatrix} w \\ \log(\Sigma_0) \\ \log(\sigma_\varepsilon^2) \end{bmatrix}; \mu_\lambda, \Sigma_\lambda \right),$$

where $\Sigma_\lambda = \text{diag}(\sigma_1^2, \dots, \sigma_K^2)$ and $\log(\Sigma_0)$ denotes element-wise application of the logarithm. For the ridge prior $\log(\Sigma_0) = \log(\tau^2)$ and for the horseshoe prior $\log(\Sigma_0) = (\log(\tau_1^2), \dots, \log(\tau_{H_{L-1}}^2))^\top$. For the ridge prior, the variational parameters have dimension $K = H_{L-1} + 2$, and for the horseshoe prior $K = 2H_{L-1} + 1$. In both cases, the computational cost of VI scales linearly in H_{L-1} , i.e. $\mathcal{O}(H_{L-1})$.

While linear scaling is attractive from a computational viewpoint, a diagonal covariance matrix implies posterior independence between the last-layer weights w , the elements of Σ_0 , and the noise variance σ_ε^2 , an assumption that is often too restrictive. On the other hand, specifying a full covariance matrix Σ_λ leads to quadratic scaling in H_{L-1} , i.e., $\mathcal{O}(H_{L-1}^2)$. As a trade-off, Ong et al. [1261] and others propose factorising the covariance matrix as

$$\Sigma_\lambda = BB^\top + D^2,$$

where $B \in \mathbb{R}^{K \times F}$ is a lower triangular, full rank matrix with $F \ll K$ and $D = \text{diag}(d_1, \dots, d_K)$ is a diagonal matrix with nonnegative entries, $d_l > 0$, $l = 1, \dots, K$.

With this variational density, $K = 2(H_{L-1} + 2) + \frac{F(F+1)}{2}$ for the ridge prior and $K = 2(2H_{L-1} + 1) + \frac{F(F+1)}{2}$ for the horseshoe prior. For both priors, VI scales with $\mathcal{O}(H_{L-1} F^2)$.

Closed-form loss

Assuming a Gaussian likelihood and a Gaussian prior, an advantage of choosing q_λ to be Gaussian is that $\mathbb{E}_{q_\lambda(w)}[\log p(y|x, w, \sigma_\varepsilon^2)]$ is of closed form, as shown in the following result by Titsias [1615]. Hence, no Monte Carlo sampling is required to obtain unbiased estimates of the expected log-likelihood during training [709].

Lemma 39.1 (Closed-form expected log-likelihood under Gaussian q_λ). *Let $q_\lambda(w) = \mathcal{N}(w; \mu_w, \Sigma_w)$ with $w \in \mathbb{R}^{H_{L-1}}$, $\mu_w \in \mathbb{R}^{H_{L-1}}$, $\Sigma_w \in \mathbb{R}^{H_{L-1} \times H_{L-1}}$, and $p(y | x, w, \sigma_\varepsilon^2) = \mathcal{N}(y; w^\top \phi, \sigma_\varepsilon^2)$ with $\phi := \phi_\theta(x) \in \mathbb{R}^{H_{L-1}}$ and $\sigma_\varepsilon^2 > 0$. Then, it holds*

$$\mathbb{E}_{q_\lambda(w)}[\log p(y | x, w, \sigma_\varepsilon^2)] = \log \mathcal{N}(y; \mu_w^\top \phi, \sigma_\varepsilon^2) - \frac{1}{2} \sigma_\varepsilon^{-2} \phi^\top \Sigma_w \phi. \quad (39.27)$$

The proof for uni- and multivariate targets ($C \geq 1$) is left as an exercise to the reader. VI is popular in practice because estimation is fast, and it is straightforward to check whether the ELBO has converged. However, a variational approximation can incur a so-called variational gap, which arises if the true posterior is not a member of the variational family, i.e., $\nexists \lambda^* : q_{\lambda^*}(w, \Sigma_0, \sigma_\varepsilon^2) = p(w, \Sigma_0, \sigma_\varepsilon^2 | \mathcal{D})$. In addition, posterior standard deviations are often underestimated [170].

39.3.3 Estimation and prediction

To train $p(w, \Sigma_0, \sigma_\varepsilon^2 | \mathcal{D})$ and θ , one can either choose a *two-step* or a *joint* approach as we will discuss next.

Two-step estimation

Alg. 32 summarises the two-step approach where θ is estimated first and used as a plug-in estimate to perform posterior estimation.

Algorithm 32: Two-step estimation

Input : training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$

Output: network parameters θ , posterior distribution $p(w, \Sigma_0, \sigma_\varepsilon^2 | \mathcal{D})$

Train the network f_θ using $\mathcal{D}$ with a standard loss function (e.g. mean squared error)

Discard last-layer weights w and freeze the remaining parameters θ

Forward pass $\{x_n\}_{n=1}^N$ through the network to obtain feature projections $\phi_n := \phi_\theta(x_n)$ for $n = 1, \dots, N$

Estimate $p(w, \Sigma_0, \sigma_\varepsilon^2 | \mathcal{D})$ using the exact posterior, MCMC, or VI

The main advantage of the two-step estimation approach is that the cost of training ϕ_θ is completely independent of the posterior approximation scheme, and any previously trained deterministic network can be extended with a last-layer inference using Alg. 32.

Joint estimation

Estimating θ and $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$ jointly takes into account the uncertainty from the last-layer weights when learning θ . For VI, we interpret the ELBO in Eq. (39.25), as a joint function of θ and λ , following a similar approach to that of Harrison et al. [709], such that

$$\text{ELBO}(\lambda, \theta) = \mathbb{E}_{q_\lambda(w, \Sigma_0, \sigma_\varepsilon^2)} [\log p(\mathcal{D} \mid \theta, w, \Sigma_0, \sigma_\varepsilon^2)] - \text{D}_{\text{KL}}(q_\lambda(w, \Sigma_0, \sigma_\varepsilon^2) \parallel p(w, \Sigma_0, \sigma_\varepsilon^2)),$$

and jointly maximise over θ, λ .

With SG-MCMC, the last-layer weights are updated according to Eq. (39.23), and the deterministic weights at iteration $t + 1$ are updated, for example, with standard SGA

$$\theta^{[t+1]} = \theta^{[t]} + \eta^{[t]} \sum_{n \in B} \nabla_\theta \log p(y_n \mid x_n, \theta^{[t]}, w^{[t]}, \Sigma_0^{[t]}, (\sigma_\varepsilon^2)^{[t]}). \quad (39.28)$$

The main drawback of joint estimation is that fitting θ and $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$ simultaneously is challenging, because changes in θ continually alter the posterior.

39.3.4 Posterior predictive distribution

For a new input x_* , the posterior predictive distribution is given by

$$p(y_* \mid x_*, \mathcal{D}, \theta) = \int p(y_* \mid x_*, \mathcal{D}, \theta, w, \Sigma_0, \sigma_\varepsilon^2) p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D}, \theta) d(w, \Sigma_0, \sigma_\varepsilon^2). \quad (39.29)$$

Assume we have point estimates $\hat{\theta}$ of the deterministic neural network weights and a sample $\{(w^{[1]}, \Sigma_0^{[1]}, (\sigma_\varepsilon^2)^{[1]}), \dots, (w^{[S]}, \Sigma_0^{[S]}, (\sigma_\varepsilon^2)^{[S]})\}$ from the joint posterior of the last-layer weights, prior parameters in Σ_0 and noise variance σ_ε^2 . The latter can be samples from the exact posterior (where available), MCMC samples, or samples from the variational approximation $q_{\lambda^*}(w, \Sigma_0, \sigma_\varepsilon^2)$. Then, Eq. (39.29) can be approximated by

$$p(y_* \mid x_*, \mathcal{D}, \theta) \approx \frac{1}{S} \sum_{s=1}^S p(y_* \mid x_*, \mathcal{D}, \hat{\theta}, w^{[s]}, (\sigma_\varepsilon^2)^{[s]}) \quad (39.30)$$

and used as an approximation to $p(y_* \mid \mathcal{D}, x_*)$ based on our assumption in Eq. (39.1).

39.3.5 Posterior predictive moments

Later in the applications, we will evaluate the posterior predictive mean

$$\mathbb{E}[Y_* \mid x_*, \mathcal{D}] = \int \mathbb{E}[Y_* \mid x_*, w, \sigma_\varepsilon^2] p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D}) d(w, \Sigma_0, \sigma_\varepsilon^2), \quad (39.31)$$

and the posterior predictive variance

$$\begin{aligned} \mathbb{V}[Y_* \mid x_*, \mathcal{D}] &= \mathbb{E}_{p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})} [\mathbb{V}(Y_* \mid x_*, w, \sigma_\varepsilon^2)] \\ &\quad + \mathbb{V}_{p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})} [\mathbb{E}(Y_* \mid x_*, w, \sigma_\varepsilon^2)], \end{aligned} \quad (39.32)$$

where y_* is the realisation of the random variable Y_* and Σ_0 can be omitted from the conditioning set due to conditional independence $Y_* \perp \Sigma_0 \mid (x_*, w, \sigma_\varepsilon^2)$. In general, it holds that $\mathbb{E}[Y_* \mid x_*, w, \sigma_\varepsilon^2]$ and $\mathbb{V}[Y_* \mid x_*, w, \sigma_\varepsilon^2]$ are the first two moments of the conditional distribution of $Y_* \mid x_*, w, \sigma_\varepsilon^2$. In our case, these are the moments of the Gaussian distribution given by $\mathbb{E}[Y_* \mid x_*, w, \sigma_\varepsilon^2] = w^\top \phi_\theta(x_*)$ and $\mathbb{V}[Y_* \mid x_*, w, \sigma_\varepsilon^2] = \sigma_\varepsilon^2$. Hence, we obtain

$$\mathbb{E}[Y_* \mid x_*, \mathcal{D}] = \int w^\top \phi_\theta(x_*) p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D}) d(w, \Sigma_0, \sigma_\varepsilon^2), \quad (39.33)$$

and by the law of total variance

$$\mathbb{V}[Y_* \mid x_*, \mathcal{D}] = \int \mathbb{V}[Y_* \mid x_*, w, \sigma_\varepsilon^2] p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D}) d(w, \Sigma_0, \sigma_\varepsilon^2) \quad (39.34)$$

$$+ \int (\mathbb{E}[Y_* \mid x_*, w, \sigma_\varepsilon^2] - \mathbb{E}[Y_* \mid x_*, \mathcal{D}])^2 p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D}) d(w, \Sigma_0, \sigma_\varepsilon^2). \quad (39.35)$$

The posterior $p(w, \Sigma_0, \sigma_\varepsilon^2 \mid \mathcal{D})$ can be integrated out numerically in practice using S samples from the posterior obtained via MCMC or VI. Therefore, the first integral is approximated using Monte Carlo integration by $\frac{1}{S} \sum_{s=1}^S (\sigma_\varepsilon^2)^{[s]}$. The second integral can be approximated by first plugging in $\mathbb{E}[Y_* \mid x_*, \mathcal{D}] \approx \frac{1}{S} \sum_{s=1}^S (w^{[s]})^\top \phi_\theta(x_*)$, and then applying Monte Carlo integration using the expression for $\mathbb{E}[Y_* \mid x_*, w, \sigma_\varepsilon^2]$ from before.

39.4 Application

We now illustrate last-layer inference on two regression tasks with synthetic and real-world data. We compare last-layer inference methods under the different prior specifications from Sec. 39.2.2 against a fully Bayesian neural network (fBNN) and MC dropout [566] (see Chapter 14 for a detailed treatment) as baselines. The last-layer inference models are trained using both exact and approximate inference. The goal is to estimate the posterior predictive density $p(y_* | x_*, \mathcal{D})$, the posterior predictive mean $\mathbb{E}[Y_* | x_*, \mathcal{D}]$, and the posterior predictive variance $\mathbb{V}[Y_* | x_*, \mathcal{D}]$, derived previously.

We conduct the analysis under three objectives. Our first objective is to check visually whether last-layer inference performs similarly to fBNN and MC dropout. All models should provide accurate posterior predictive means for the targets, capture uncertainty within the training range, and yield higher uncertainty at input points that lie far from the training data, so-called “out-of-distribution”, where the model is not expected to generalise.

In the second objective, we inspect the coverage rates of the prediction intervals. Following the definition of Pearce et al. [1305], a $1 - \alpha$ prediction interval is an interval $[\hat{y}_{n,\text{LB}}, \hat{y}_{n,\text{UB}}]$, such that at least a proportion $1 - \alpha$ of the observations fall within it

$$\mathbb{P}[\hat{y}_{n,\text{LB}} \leq y_n \leq \hat{y}_{n,\text{UB}}] \geq 1 - \alpha. \quad (39.36)$$

The coverage rates of the prediction intervals should be accurate so that the width of the prediction interval at a given level can be used to quantify uncertainty of the current prediction. In autonomous driving applications, for example, the width of a prediction interval for the steering angle at a prespecified prediction level can be used to perform model selection or switch to manual control [765].

Finally, in the third objective, we evaluate how well the posterior predictive mean fits the observed data, measured by the predictive root mean square error (RMSE). Let $\hat{y}_n$ for $n = N + 1, \dots, N^*$ denote the posterior predictive mean of observation n in the test set, where N^* denotes the combined number of observations in the training and test set. The RMSE is then

$$\text{RMSE} = \sqrt{\frac{1}{N^* - N} \sum_{n=N+1}^{N^*} (\hat{y}_n - y_n)^2}, \quad (39.37)$$

where y_n is the respective observed target value.

39.4.1 Datasets

First, we consider a synthetic dataset generated by a noisy sine function

$$y = g(x) = \sin(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \quad (39.38)$$

where $\sigma_\varepsilon = 0.3$. To create a training and validation set, the function $g(x)$ is evaluated at 200 evenly spaced input values in the interval $[-4, 4]$. From these 200 samples, 100 are randomly selected as the training set and validation set with an 80/20 split. The remaining 100 samples form the test set.

As real-world data, we use the concrete compressive strength dataset [1814] from the UCI Machine Learning repository [878]. The goal is to predict how well the concrete can withstand compressive forces based on eight features, including the age of the concrete and its water content. The dataset comprises 1,038 samples. From these, 830 samples are used for training (from which 20 percent are used for validation during training) and 208 samples for testing. Both the features and targets are standardised using the training set statistics.

39.4.2 Implementation details

While most estimation methods for last-layer inference admit jointly fitting the last-layer weights and parameters of the deep feature projector ϕ_θ , we focus on two-step estimation throughout. This is the most common approach in practice, since it allows uncertainty estimation to be added to any deterministic neural network after training. The hyperparameters, architectures, and training settings for each model are as follows.

Last-layer inference

The network used as a basis for the deep feature projector ϕ_θ is a feedforward network with dense layers and ReLU activations in all layers except the output layer. For the synthetic data, the network has dimensions

[1, 100, 100, 100, 10, 1], and for the concrete compressive strength data [8, 200, 200, 100, 10, 1]. It is trained for 100 epochs with the Adam optimiser [894] and a learning rate of 10^{-3} .

We perform last-layer inference with three different prior specifications: (1) a Gaussian prior on the weights combined with a Gaussian distribution for the observational noise and a fixed noise variance σ_ε^2 , (2) the ridge prior, and (3) the horseshoe prior. For (1), we choose $\tau^2 = 1$ and $\sigma_\varepsilon = 0.3$. For (2) and (3), the noise is inverse gamma distributed $\sigma_\varepsilon^2 \sim \mathcal{IG}(a_\sigma, b_\sigma)$ with $a_\sigma = b_\sigma = 2$. For the ridge prior, we set $a_\tau = b_\tau = 2$. For the horseshoe prior, we set $\xi = 1$ to simplify computation, as inference over the global shrinkage parameter led to challenging posterior geometries in our case. This simplified horseshoe prior can provide local shrinkage but does not adapt the overall shrinkage level to the observed data. Alternative approaches for specifying and calibrating the prior for the global shrinkage parameter are discussed in Piironen and Vehtari [1325]. For each prior choice, we use both MCMC and VI with the following settings.

We employ different MCMC algorithms for each last-layer specification. We fit (1) with SG-MCMC. This is an attractive solution if there is a large number of observations, such that computing a closed-form posterior is computationally infeasible, but inference should be exact. We use the settings from Welling and Teh [1737], where the learning rate η is decayed from 0.1 to 0.001 during training via $\eta^{[t]} = a(b+t)^{-\gamma}$ with $\gamma = 0.55$ and appropriate choices for a and b , and where $t \geq 0$ denotes the iteration. The posterior predictive mean is computed based on a weighted average using the last S samples

$$\mathbb{E}[Y_* \mid x_*, \mathcal{D}] \approx \frac{\sum_{s=1}^S \eta^{[s]} \phi_\theta(x_*) w^{[s]}}{\sum_{s=1}^S \eta^{[s]}}. \quad (39.39)$$

as proposed in Welling and Teh [1737]. We run SG-MCMC for 2,000 iterations and discard the first 1,000 samples as burn-in. For (2), we employ a Gibbs sampler, which is run for 2,000 iterations, and the first 1,000 samples are discarded as burn-in. For the Gibbs sampler, the posterior predictive distribution is a Gaussian. Since (3) is a more complex model, we use HMC, which is known to be faster and less sticky than samplers based on random walks [1229]. We use different settings for each dataset, which we indicate in parentheses. We run the HMC sampler for 5,000 (*synthetic*) and 8,000 (*concrete*) iterations with 18 (*synthetic*) and 20 (*concrete*) leapfrog steps, and a step size of 0.0045 (*synthetic*) and 0.0015 (*concrete*). The parameters and number of iterations were chosen so that trace plots suggested convergence, and the HMC acceptance rate was about 65 percent. For the ridge prior with a Gibbs sampler, the effective sample size [ESS; 915] averaged across all weight dimensions is 958 (*synthetic*) and 1,939 (*concrete*), with a Gelman-Rubin diagnostic [600] of $\hat{R} \approx 1.03$ for both datasets. For the horseshoe prior with HMC, the chain is not well-mixed $\hat{R} > 1.5$, and the effective sample sizes are low (≈ 125). This highlights the challenge of choosing an appropriate MCMC scheme for more complex model specifications. Convergence may be improved by using more sophisticated samplers such as the No-U-Turn Sampler (NUTS) [762].

For VI, we use the three previously introduced schemes. We employ the closed-form expression from Lemma 39.1 for the Gaussian with a full covariance matrix in specification (1). For the ridge (2) and horseshoe (3) cases, we use a Gaussian variational distribution with both diagonal and factored covariance matrices, where for the latter we heuristically choose $F = 5$ (ridge) and $F = 15$ (horseshoe) factors. A more precise approach would be to perform a grid search and choose the number of factors that yield the best predictive performance on a held-out validation set. For all VI schemes, the learning rate is 10^{-3} , and we run the schemes until the ELBO has converged. This is checked visually, though automatic stopping criteria could also be employed. This results in 1,000 iterations for prior specification (1), 20,000 iterations for (2), as well as 30,000 (diagonal), and 50,000 (factored) iterations for (3).

MC dropout

For MC dropout, we use the same architecture and training hyperparameters as before. In addition, a so-called dropout probability p_{drop} has to be specified, which defines the probability of each neuron in the network being “dropped out” during training and prediction. In other words, certain weights and biases are randomly set to zero. During training, this acts as regularisation and thereby prevents overfitting. During prediction, each forward pass under a different dropout mask can be interpreted as a sample of the posterior predictive distribution. We choose $p_{\text{drop}} = 0.2$ based on [566].

Fully Bayesian neural network

For the fBNN, we place a Gaussian prior with mean 0 and variance 0.1 on each weight and bias. This prior variance is a common default choice, but one could also determine it by grid-search or scale the prior variance by the dimension of the input [1226]. The corresponding posterior is approximated with mean-field VI (see

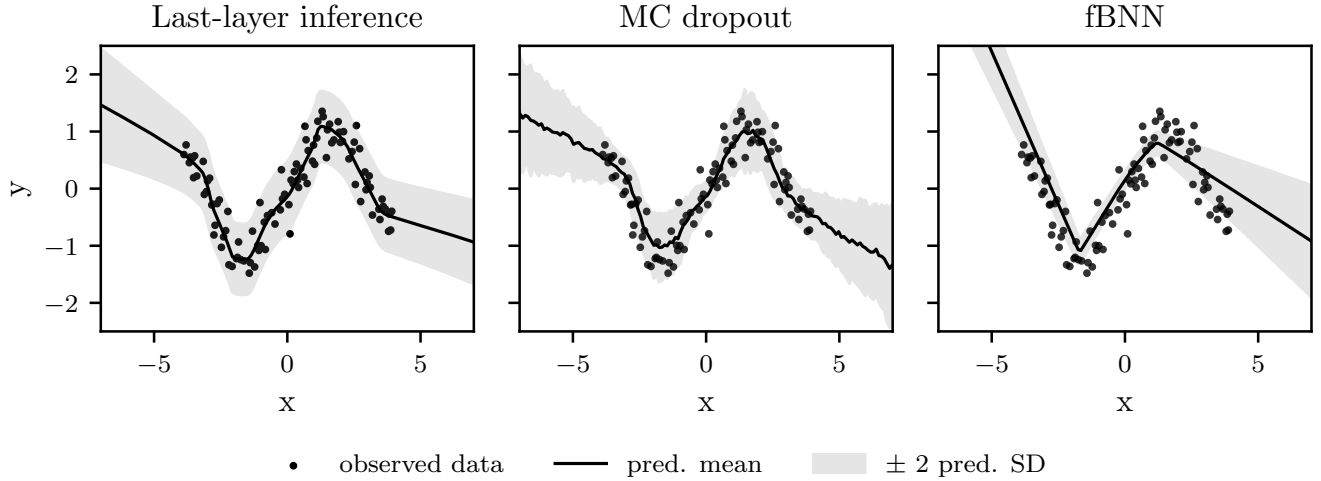


Figure 39.3: Comparison of last-layer inference with a Gaussian prior and closed form predictive posterior distribution (left), MC dropout (centre), and an fBNN (right) fitted to 100 points from the noisy sine function in Eq. (39.38).

Chapter 13) using the Bayes-by-Backprop algorithm by Blundell et al. [174] via the `blitz` package in Python [498], a KL weight of 10^{-3} , three MC samples per training step, and a learning rate of 10^{-2} . The fBNN is trained for 200 epochs and has hidden dimensions [100, 100] (synthetic) and [150, 150] (concrete) with ReLU activation in all layers except in the output layer.

All models are trained on the same training data using a batch size of 32 (synthetic) and 64 (concrete).

39.4.3 Results

Visualisation on synthetic data

Fig. 39.3 illustrates the posterior predictive means and pointwise intervals corresponding to two posterior predictive standard deviations on the synthetic dataset. We compare three models: the last-layer inference specification with a Gaussian prior (1; left), where the exact posterior is available in closed form, MC dropout (middle), and an fBNN (right).

Overall, all models capture the underlying sine function well in terms of the posterior predictive mean. MC dropout produces jagged standard deviation intervals across the input space, whereas last-layer inference and the fBNN yield smoothly varying intervals. For all models, predictive uncertainty increases for input values far from the training data, highlighting their potential to detect out-of-distribution observations. The visualisations for the ridge and horseshoe prior appear virtually identical to last-layer inference with the Gaussian prior and are therefore omitted.

Coverage of prediction intervals

Next, we inspect the coverage rates of the prediction intervals. Fig. 39.4 and Fig. 39.5 show the difference between the predicted and empirical coverage of the prediction intervals. We compare several last-layer approaches, as well as the fBNN and MC dropout. The x -axis measures the level of the prediction interval, and the y -axis measures the average difference between the predicted coverage and empirical coverage on the test set. A reliable probabilistic model would produce a horizontal line at a value of 0 on the y -axis, indicating that the predicted and empirical coverage coincide. If, at a given level, a model produces a value below zero, it is underconfident. Vice versa, if the value is greater than zero, it is overconfident. In Fig. 39.4, last-layer inference with a closed-form Gaussian predictive distribution (1; solid line), MC dropout (dashed line), and fBNN (dash-dotted line) produce similar coverage rates. On average, last-layer inference produces comparable or less reliable coverage for levels up to around 85 percent (synthetic) and 70 percent (concrete). For higher levels, last-layer inference performs better. Across all levels, the models predict too-narrow prediction intervals, resulting in overconfidence. In the worst case, the predicted coverage is up to 20 percentage points higher than the empirical coverage. Fig. 39.5 compares the coverage rates for last-layer inference with different prior specifications. Coverage varies considerably between models fitted using VI and MCMC. For clarity, we only show the models with the highest average coverage for

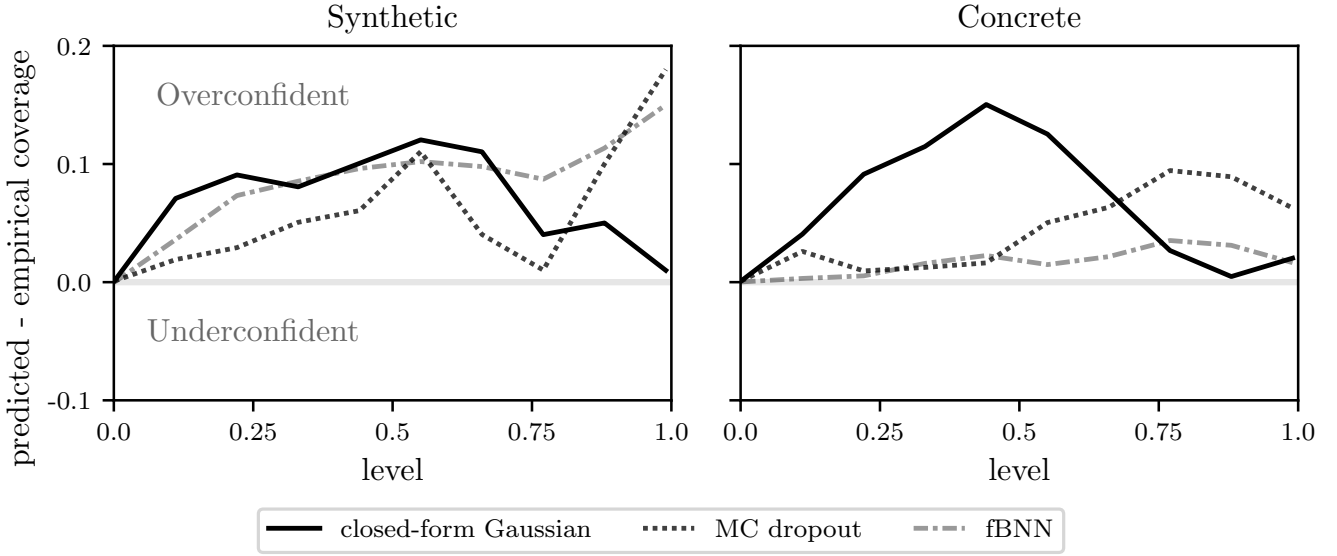


Figure 39.4: Prediction interval coverage rates for last-layer inference, fBNN and MC dropout on the test set for the noisy sine function from Eq. (39.38) (left) and concrete compressive strength data (right).

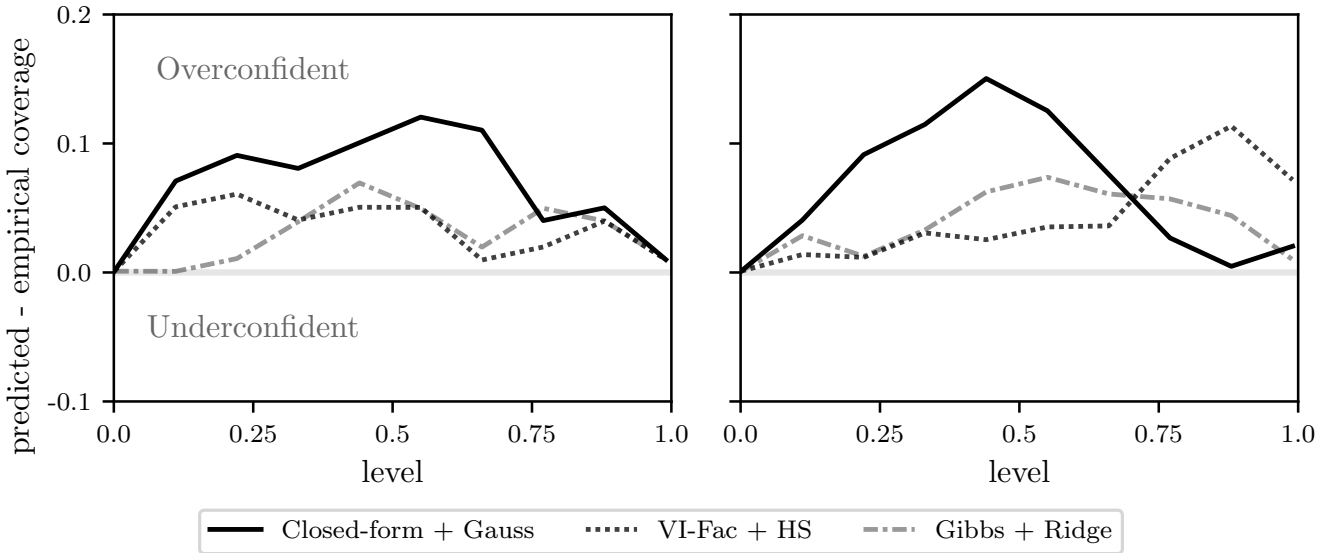


Figure 39.5: Prediction interval coverage rates for last-layer inference with different priors on the test set for the noisy sine function from Eq. (39.38) (left) and concrete compressive strength data (right).

each prior. For the synthetic data, the ridge (2; light dash-dotted line) and horseshoe priors (3; dotted line) deliver better coverage rates on average compared to a fixed variance Gaussian prior. This illustrates that a more sophisticated Bayesian treatment via a hierarchical model can be beneficial to obtain accurate uncertainty quantification. For the concrete data, the ridge and horseshoe settings provide more reliable coverage for levels up to 70 percent, and for higher levels, the closed-form Gaussian setting performs best.

Accuracy of posterior predictive means

To quantify predictive accuracy, Table 39.1 shows the RMSE on the test set for each method. Overall, the performance is similar across methods, with fully Bayesian approaches achieving the best performance. Among all last-layer inference models on the synthetic data, exact inference using SG-MCMC with a Gaussian prior results in the lowest RMSE, yet performs slightly worse than the fBNN. On the concrete data, the ridge prior combined with a Gibbs sampler performs best among the last-layer inference models, and MC dropout achieves the lowest RMSE overall. If the output layer is low-dimensional, the last-layer inference models are computationally cheaper during

prediction, because the posterior predictive mean can be computed analytically or numerically using posterior samples. In contrast, MC dropout and the fBNN require several stochastic forward passes.

Table 39.1: RMSE on the test set for different last-layer inference methods, fBNN and MC dropout on the noisy sine function from Eq. (39.38) and the concrete compressive strength data. Bold indicates the best RMSE across all methods per dataset.

Category	Type	Estimation	Prior	Synthetic	Concrete
Last-layer inference	exact	Closed-form	Gaussian	1.03	1.43
		SG-MCMC	Gaussian	0.98	1.50
		HMC	horseshoe	0.99	1.43
		Gibbs	ridge	1.02	1.42
	approx.	VI closed-form	Gaussian	1.03	1.43
		VI-Diag	ridge	1.02	1.43
		VI-Diag	horseshoe	1.03	1.43
		VI-Fac	ridge	1.03	1.43
		VI-Fac	horseshoe	1.02	1.43
	Fully Bayesian	MC dropout	–	0.98	1.37
		fBNN	–	0.96	1.39

For the horseshoe and ridge priors, employing a full covariance matrix (VI-Fac) instead of a diagonal covariance matrix (VI-Diag) in the VI scheme does not lead to a substantial improvement in RMSE. While last-layer inference with SG-MCMC produces a low RMSE for the synthetic data, it exhibits convergence issues for the concrete data, resulting in poor fits and large RMSE compared to the other methods. This highlights the challenge of tuning the hyperparameters for (SG-)MCMC samplers in more complex scenarios.

Fig. 39.6 shows the posterior means and standard deviations for the last-layer weights w comparing exact with approximate Bayesian inference. Points on the diagonal indicate that the exact and approximate estimates coincide. Generally, both VI schemes provide good approximations for the posterior predictive means, with a few outliers for the horseshoe prior. The posterior predictive standard deviations suffer from over- and underestimation. This is particularly the case for the horseshoe specification (3), although this may reflect convergence issues with the HMC sampler rather than misestimation.

Impact of the feature projector’s training procedure

Lastly, we look at the influence of the training procedure of the deep feature projector. Fig. 39.7 illustrates the posterior predictive mean and prediction intervals of last-layer inference for the specification with a ridge prior (2) with three different training regimes for the deep feature projector ϕ_θ . Each feature projector was trained with a mean squared error loss and an ℓ_2 regularisation on the network weights and biases, that is $\alpha/2 \cdot \|\vartheta\|_2^2$, where α controls the strength of the regularisation. This regularisation can be interpreted as a Gaussian prior with variance α^{-1} (compare ℓ_2 regularisation priors in Sec. 28.1.3). A high value of α , therefore, is equivalent to smaller prior variances when training ϕ_θ .

Fig. 39.7 illustrates that introducing a wider prior over the last-layer weights during the second estimation step cannot compensate for having previously trained the deep feature projector with strong weight regularisation. With increasing ℓ_2 regularisation (left to right), the entries of the posterior covariance matrix Σ_N remain close to zero. The effects of this are visible in the left and middle panels, where the predictive uncertainty does not vary with x .

39.5 Extensions and limitations

We now discuss some general properties, extensions, and further applications of last-layer inference.

39.5.1 Beyond canonical Bayesian linear regression models

Some last-layer models can capture multimodal predictive densities and ensure specific calibration properties. For example, Klein et al. [902] propose marginally calibrated deep distributional regression, which is a last-layer

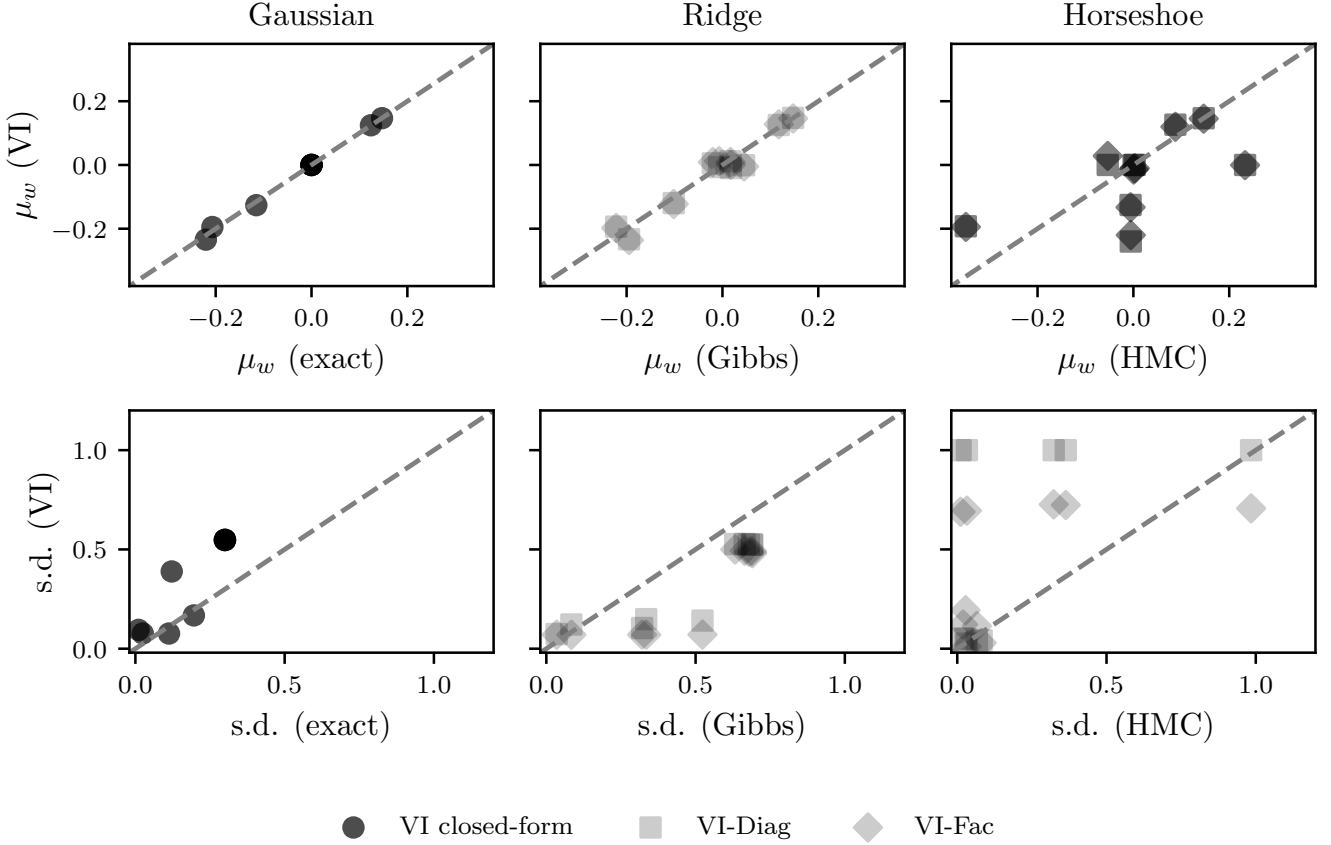


Figure 39.6: Comparison of the posterior means (top) and standard deviations (bottom) of w with exact and approximate estimation for different prior specifications (columns) with the synthetic data from Eq. (39.38).

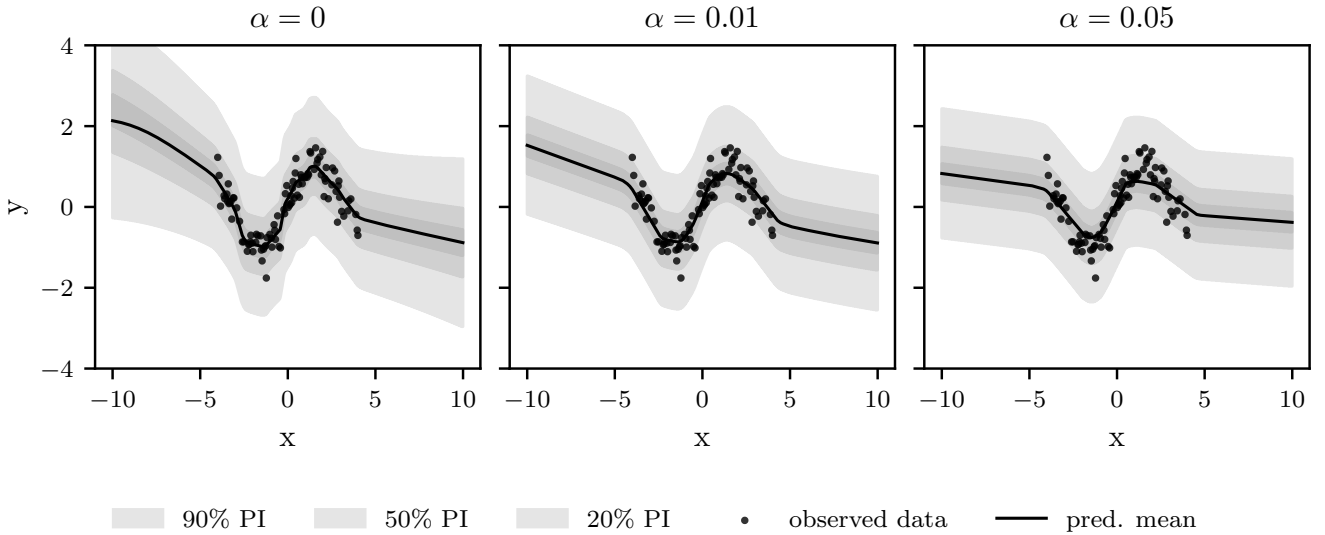


Figure 39.7: The effect of the ℓ_2 regularisation weight α used to train the deep feature projector ϕ_θ on predictive uncertainty of last-layer inference in a two-step estimation using the noisy sine function from Eq. (39.38).

inference approach based on a variable transformation model and a conditional Gaussian copula. This model produces flexible predictive densities exhibiting marginal calibration [640], that is, the asymptotic limits of the true marginal and average predictive distributions coincide. This approach was leveraged in Hoffmann and Klein [765] for autonomous driving, where training sets are too large to fit fBNNs.

Last-layer inference can also be extended beyond the standard Bayesian linear regression assumptions. In

Eq. (39.4), we assumed homoscedastic observational noise with $\varepsilon_n \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, where the variance σ_ε^2 does not vary with the inputs x . In practice, the noise variance may depend on the inputs (heteroscedasticity). If this is expected, last-layer inference can be extended by allowing the noise variance to vary with the inputs. Harrison et al. [710] propose efficient estimation techniques for this model specification using VI.

39.5.2 Further applications

Last-layer inference has proven useful in several applied settings. One important example is Bayesian optimisation, which aims to optimise a target function that is expensive to evaluate. To avoid direct computation, a surrogate model approximates the target function, and optimisation is performed with the surrogate model instead. A good surrogate model quantifies uncertainty well and is cheap to evaluate. Last-layer inference fits both of these criteria and has proven competitive to other methods in benchmark studies [214, 1530].

In addition, last-layer inference has also been successfully applied to reinforcement learning, particularly in the widely studied contextual bandits problem (see Chapter 56). Riquelme et al. [1397] demonstrate that last-layer inference performs competitively with more complex Bayesian neural network approaches.

39.5.3 Drawbacks of last-layer inference

Last-layer inference becomes less effective when the output layer is high-dimensional. This situation arises, for instance, in image segmentation tasks with U-Net [1419] architectures, where the output dimension is high. In such cases, a more suitable approach is to perform Bayesian inference on an intermediate, lower-dimensional layer.

Finally, last-layer inference is not directly applicable to meta-learning frameworks such as the neural process family [582] because these models require task-dependent weight posteriors.

39.6 Final remarks

Last-layer inference is a powerful and computationally cheap method to estimate posterior predictive distributions with neural networks that can, in certain situations, produce similar performance to fully Bayesian approaches. There is a vast selection of Bayesian linear regression models that can be transformed into last-layer inference models, and the performance can be improved by employing appropriate prior structures.

Chapter 40

Scalable Laplace approximations

The previous chapters have established the linearised Laplace approximation as a principled and powerful framework for equipping neural networks with posterior uncertainty. By linearising the network around a point estimate (setting $\theta_0 \equiv \theta_*$ in Equation 8.1 of Chapter 8), we arrived at a Gaussian posterior $p(\theta \mid \mathcal{D}) \approx \mathcal{N}(\theta; \theta_*, \Psi^{-1})$ where the curvature Ψ from Equation 8.4 is often decomposed into the Generalised Gauss-Newton (GGN) matrix from Equation 8.22 along with the Hessian of the regulariser term. However, modern deep learning confronts us with formidable quantities in the network Jacobians and Hessians. The curvature matrix $\Psi \in \mathbb{R}^{P \times P}$ has a memory footprint of $\mathcal{O}(P^2)$ and requires $\mathcal{O}(P^3)$ computational effort for inversion if performed naively. Similarly, even if we approach the intractability of the curvature matrix by treating it as a linear operator, predictive posteriors and marginal likelihoods of Laplace approximations can be intractable or require instantiations of network Jacobians, which are matrices in $\mathbb{R}^{C \times P}$. For modern deep neural networks, where the number of parameters P can easily be in the millions or billions, the direct instantiation and manipulation of this matrix is utterly infeasible. In this chapter, we will address this challenge head-on by introducing a suite of techniques designed to make Laplace approximations tractable for large-scale models, making them a practical tool for the modern deep learning practitioner. We will explore three main avenues: (i) approximating the GGN matrix G with structured, computationally friendly alternatives; (ii) exploiting the function-space duality with Gaussian processes to perform inference over a reduced set of datapoints or inducing points; and (iii) sidestepping the explicit computation of the posterior covariance altogether, drawing posterior samples via iterative linear solvers. Table 40.1 summarises the methods covered in this chapter, organised by the space in which each approximation is made. Across these avenues, we must be careful to ensure that we maintain a principled approach to hyperparameter estimation using the marginal likelihood, and we will discuss these challenges in more detail.

40.1 Structured Approximations to the Curvature

The core difficulty of the Laplace approximation lies in the sheer size and lack of structure of the curvature matrix Ψ , which is mostly due to the GGN matrix component. The most direct path to scalability is therefore to replace G with an approximation $\tilde{G}$ that is easier to store, invert, and perform downstream calculations with. We briefly discussed some strategies in Section 8.4, and we expand upon them in the following section.

Table 40.1: Overview of the scalable Laplace approximations covered in this chapter, organised by the space in which the approximation is made.

Method	Space	Key approximation	Dominant cost
Diagonal	Parameter	Off-diagonal curvature = 0	$\mathcal{O}(P)$
KFAC	Parameter	Layer-wise Kronecker blocks	$\sum_i \mathcal{O}((D^{(i)})^3)$
Last-layer / subnetwork	Parameter	Dirac priors on θ_D	$\mathcal{O}(P_B^3)$
MoE LLA	Function	Disjoint data clusters	$\sum_i \mathcal{O}((N_i C)^3)$
VaLLA	Function	M inducing points	$\mathcal{O}(NM^2 + M^3)$
STO with CG/SGD	Sampling	Iterative solvers, MC predictive	$\mathcal{O}(KPN)$ per sample
Sampling-based EM	Sampling	Stochastic trace estimation	Matrix-vector products

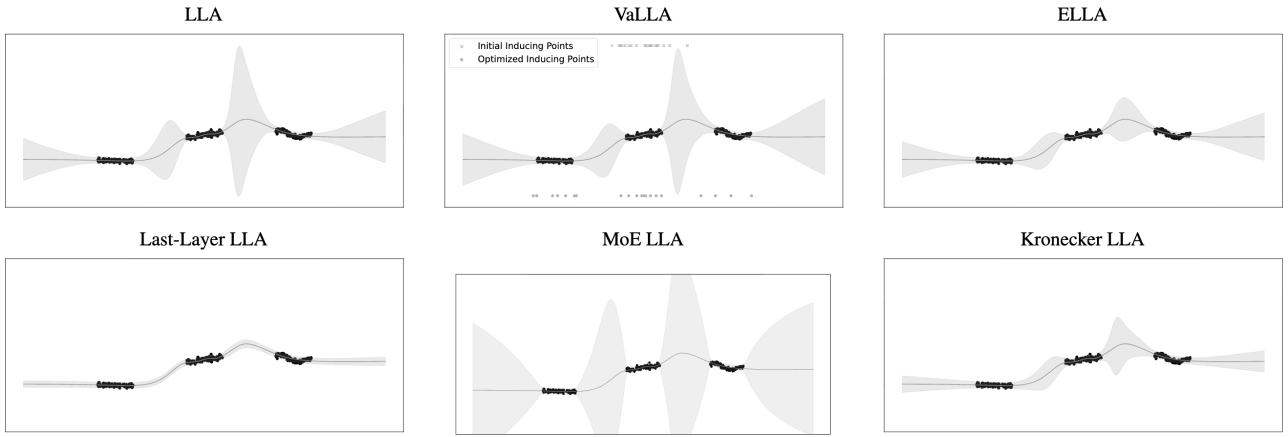


Figure 40.1: Predictive distributions (mean as a solid line, with the shaded region denoting two standard deviations) on a toy 1D regression dataset with a two-hidden-layer MLP with 50 units trained using back-propagation. The predictive distribution of the variational linearised Laplace approximation (VaLLA) is on par with or better than the last-layer and Kronecker-factored approximations, a mixture-of-experts LLA (MoE LLA) with 200 clusters, and the efficient low-rank approximation ELLA. The optimal values for the noise and prior are obtained by maximising the marginal log-likelihood estimate of the full LLA; VaLLA and ELLA reuse these values. VaLLA uses 20 inducing points for the predictive variances; ELLA uses 20 random locations and 20 features. Figure reproduced with permission from Ortega et al. [1265].

40.1.1 Diagonal approximations

The most drastic simplification is to assume that all off-diagonal elements of the curvature matrix are zero, i.e. $\tilde{G} = \text{diag}(G)$, which is equivalent to asserting that all parameters are independent *a posteriori*. The appeal of this method is its extreme computational efficiency. Inverting a diagonal matrix is a trivial operation with $\mathcal{O}(P)$ complexity, as it simply involves taking the reciprocal of each diagonal element. However, this efficiency comes at a significant cost in expressiveness. The assumption of parameter independence is strong and often unrealistic, ignoring the complex correlations that are learned during training. While it can be a useful baseline, the diagonal approximation often underestimates uncertainty in directions of strong parameter covariance, potentially leading to overconfident predictions [380].

40.1.2 Kronecker Factorisations

A more sophisticated approach seeks a middle ground between the full-rank GGN and the overly simplistic diagonal approximation, by assuming independence *between* layers while retaining correlations between parameters *within* layers. The Kronecker-factored approximation, often known as Kronecker-Factored Approximate Curvature (KFAC), exploits the layered structure of neural networks to construct a block-diagonal approximation of the GGN matrix. Reminding ourselves of the notation we previously established in Section 8.4, for a fully-connected neural network with $i = 1, \dots, L$ layers with corresponding weights $W^{(i)} \in \mathbb{R}^{D_{\text{out}}^{(i)} \times D_{\text{in}}^{(i)}}$, the KFAC approximation represents each layer-wise block of the GGN matrix $G^{(i)}$ with a Kronecker product of two lower-dimensional matrices $\Phi^{(i)}$ and $B^{(i)}$, giving us

$$\tilde{G}^{(i)} \approx \Phi^{(i)} \otimes B^{(i)}, \quad \Phi^{(i)} \in \mathbb{R}^{D_{\text{in}}^{(i)} \times D_{\text{in}}^{(i)}}, \quad B^{(i)} \in \mathbb{R}^{D_{\text{out}}^{(i)} \times D_{\text{out}}^{(i)}}. \quad (40.1)$$

The computational payoff comes from the identity $(\Phi^{(i)} \otimes B^{(i)})^{-1} = (\Phi^{(i)})^{-1} \otimes (B^{(i)})^{-1}$: by assuming this Kronecker-factored structural independence between the input and output sides of each layer, the inversion of a $D_{\text{in}}^{(i)} D_{\text{out}}^{(i)} \times D_{\text{in}}^{(i)} D_{\text{out}}^{(i)}$ block reduces to two far smaller inversions of cost $\mathcal{O}((D_{\text{in}}^{(i)})^3)$ and $\mathcal{O}((D_{\text{out}}^{(i)})^3)$. For each layer of the neural network, the Kronecker form is easy to store and perform curvature-vector products with, without ever needing to instantiate the dense $P \times P$ matrix; refer to Dangel et al. [367] for more details.

40.1.3 Inference over a subset of parameters

An alternative strategy for scalability is not to approximate the curvature matrix itself, but to reduce the number of parameters over which we perform full Bayesian inference. We partition the full parameter vector θ into a “Bayesian” subset θ_B and a “deterministic” subset θ_D . We then apply the Laplace approximation only to θ_B , treating θ_D as fixed at their point-estimate values. This is equivalent to placing an infinitely narrow (Dirac-delta) prior on the deterministic parameters. The posterior is then approximated as

$$p(\theta \mid \mathcal{D}) \approx p(\theta_B \mid \mathcal{D}) \prod_{\theta_d \in \theta_D} \delta(\theta_d - \theta_{d,*}), \quad (40.2)$$

where $p(\theta_B \mid \mathcal{D})$ is a Gaussian obtained from a Laplace approximation over the sub-space of θ_B . This idea is termed *variance pruning* by Daxberger et al. [381], since it effectively sets the variance of the deterministic set of parameters to zero, while maintaining the predictive capability of the model and attempting to closely approximate the predictive uncertainty of the full-scale Laplace approximation. Here, we discuss a few of these methods in further detail.

Last-Layer Laplace Approximations

As we established in Chapter 39, the last layer of neural networks can often be modelled probabilistically while treating the preceding layers as a fixed, non-linear feature extractor. Since the final layer is often a linear transformation from the dimensionality of the penultimate layer to the output dimensions ($W^{(L)} \in \mathbb{R}^{C \times D_{\text{in}}^{(L)}}$), the number of parameters P_L is much smaller than the total number of parameters P , drastically reducing the cost of the Laplace approximation from $\mathcal{O}(P^3)$ to a manageable $\mathcal{O}(P_L^3)$.

Subnetwork Inference

A more general approach to selecting a subset of weights to model probabilistically involves finding a subnetwork $\theta_S \subseteq \theta$, of size $P_S = |\theta_S|$, that produces the best approximate predictive posterior to that of the original neural network, but with a fraction of the weights. For making predictions themselves, i.e. the predictive mean, we still use the full neural network output $f(x, \theta)$ as before. While reasoning in the function space of the model outputs can be hard, it is easier to calculate discrepancy metrics in the weight space directly. Unfortunately, since we place Dirac delta distributions over the deterministic subset of weights, we cannot use common discrepancy metrics such as the KL divergence. Daxberger et al. [381] use the squared 2-Wasserstein distance instead, which remains well-defined between distributions without common support (unlike the KL divergence). We model the posterior distribution over the subnetwork weights using a full-covariance GGN-Laplace approximation of the form

$$p(\theta_S \mid \mathcal{D}) \approx \mathcal{N}(\theta_S; \theta_{S,*}, \Psi_S^{-1}), \quad \Psi_S \in \mathbb{R}^{P_S \times P_S}, \quad (40.3)$$

where the GGN component of Ψ_S is given by

$$G_S = \sum_{n=1}^N J_{\theta_S} f(x_n, \theta)^\top \underbrace{\nabla_z \nabla_z^\top \ell(z, y_n) \big|_{z=f(x_n, \theta)}}_{=: H_n} J_{\theta_S} f(x_n, \theta). \quad (40.4)$$

Fortunately, the squared 2-Wasserstein distance between two Gaussian distributions is available in closed form, and by making a further assumption of independence between each parameter, we recover a simplified objective

$$W_2^2 \left(p(\theta \mid \mathcal{D}), p(\theta_S \mid \mathcal{D}) \prod_{\theta_d \in \theta_D} \delta(\theta_d - \theta_{d,*}) \right) = \sum_{i=1}^P \sigma_i^2 (1 - m_i), \quad \sigma_i^2 = [\Psi^{-1}]_{ii}, \quad (40.5)$$

where $m_i = 1$ if $\theta_i \in \theta_S$ and 0 otherwise. The objective in Equation 40.5 is trivially minimised by picking a subnetwork with the P_S highest marginal variances, which are easily approximated by a diagonal approximation to Ψ . While it might seem counter-intuitive that we must resort to diagonal approximations of the Hessian again, it is important to note that the diagonal assumption is made solely for the purpose of subnetwork selection, and we still perform full covariance inference over θ_S , allowing the rich correlations between parameters in the chosen subset to contribute meaningfully to the predictive uncertainty.

40.1.4 Inference over a subset of datapoints

In Chapter 8, we established that the linearised Laplace approximation turns any differentiable neural network into a Gaussian process (GP), with an induced Mercer kernel of the form

$$\hat{k}(x, x') = J_\theta f(x, \theta_*) \Psi(\theta_*)^{-1} J_\theta f(x', \theta_*)^\top, \quad \hat{k}(x, x') \in \mathbb{R}^{C \times C}, \quad (40.6)$$

where the Jacobian pushes the parameter-space uncertainty captured by Ψ into the function space. Note that for networks with multiple outputs the kernel is matrix-valued; scalar-kernel expressions in what follows extend blockwise. If we make an assumption about the prior distribution $p(\theta) = \mathcal{N}(0, A^{-1})$ for a prior precision matrix A , we can also write the induced Mercer kernel of the prior GP distribution as

$$k(x, x') = J_\theta f(x, \theta_*) A^{-1} J_\theta f(x', \theta_*)^\top. \quad (40.7)$$

In function space, while working with GPs, it can be easier to consider quantities in the form of matrices, allowing us to represent tedious nested sums as matrix multiplications and simultaneously letting us collect quantities calculated over datasets $\mathcal{D}$ into matrices that imply their dimensionality more clearly. Here, we define the matrix $J \in \mathbb{R}^{NC \times P}$ as the vertical stacking of the per-datapoint Jacobians, $J = [J_\theta f(x_1, \theta)^\top, \dots, J_\theta f(x_N, \theta)^\top]^\top$, and the Hessian of the loss $H \in \mathbb{R}^{NC \times NC}$ as the block-diagonal matrix where each block is given by $H_n = \nabla_z \nabla_z^\top \ell(z, y_n)|_{z=f(x_n, \theta)}$. With this notation, the posterior covariance follows from applying the Woodbury matrix identity to the parameter-space posterior precision $\Psi = G + A = J^\top H J + A$: inverting Ψ in the P -dimensional parameter space is exchanged for inverting an $NC \times NC$ matrix in function space,

$$\hat{k}(x, x') = k(x, x') - k(x, X) \underbrace{(H^{-1} + J A^{-1} J^\top)^{-1}}_{:=Q^{-1}} k(X, x'). \quad (40.8)$$

The function-space duality in Equation 40.8 allows us to easily reason about the linearised Laplace approximation as an equivalent GP for all intents and purposes, letting us trade off the cubic complexity in the parameter space with a cubic complexity in the number of datapoints times the number of output dimensions instead, which is often intractable itself in modern machine learning. In this section, however, we discuss a few approaches to circumvent this complexity by exploiting the rich literature on scalable Gaussian processes (see Chapter 38 for a dedicated treatment).

Mixture of Experts Laplace Approximations

As a simple approach, explored in Lee et al. [1002], we could consider creating disjoint subsets of the training data $\{\mathcal{D}_1, \dots, \mathcal{D}_M\}$ with $\cup_{i=1}^M \mathcal{D}_i = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$. For each of these subsets, we can fit individual GPs that would have a cubic complexity in $N_i C \ll NC$. This is a simple idea that is further improved by choosing datasets such that points within each cluster are highly correlated and clusters themselves have low correlation, and also incorporating the predictions of neighbouring GPs into the prior in order to make predictions on points at the border of clusters.

Variational Linearised Laplace Approximations

Instead of picking disjoint smaller subsets of the training data and pooling predictions from multiple GPs, we can follow the approach of variational GPs, which approximate the GP posterior using a small set of M inducing points Z with their associated function-space predictions $u = f(Z, \theta)$, following the sparse variational Gaussian process (SGP) framework introduced in Chapter 38. This is precisely the strategy adopted by Variational Linearised Laplace Approximations (VaLLA) [1265]. VaLLA bridges the gap between the linearised Laplace approximation and the SGP framework. The core idea is to construct a low-rank approximation to the GP posterior using M inducing points, where $M \ll N$. A key insight of VaLLA is to use a *decoupled* SGP parameterisation, which separates the treatment of the posterior mean and covariance. Given that the Laplace approximation is already centred at a high-quality point estimate, the MAP solution θ_* , VaLLA anchors its posterior mean function to the neural network's output, $f(x, \theta_*)$. It then uses the M inducing points solely to build a tractable, low-rank approximation of the posterior covariance. The parameters of this covariance (the inducing locations Z and an associated variational precision matrix V) are optimised by maximising a variational objective that drives the sparse approximation towards the true, intractable GP posterior; as we discuss in Section 40.3, VaLLA uses an α -divergence objective in place of the standard ELBO. This optimisation has a cost of $\mathcal{O}(NM^2 + M^3)$, which is significantly more manageable than the $\mathcal{O}((NC)^3)$ cost of the full function-space GP. Figure 40.1 compares the

resulting predictive distributions against other scalable approximations on a toy regression task. The resulting predictive covariance from VaLLA, given in Equation 40.9, takes a standard form from the sparse GP literature:

$$\hat{k}_{\text{VaLLA}}(x, x') = k(x, x') - k(x, Z) (V^{-1} + k(Z, Z))^{-1} k(Z, x'), \quad (40.9)$$

where $k(x, x')$ is the prior kernel from Equation 40.7, $k(Z, Z)$ is the $M \times M$ prior kernel matrix evaluated at the inducing points, and V is a variational matrix optimised during training. While VaLLA is formulated in function space, it can also be interpreted as defining an implicit, low-rank approximation to the parameter-space curvature matrix Ψ . The full curvature $\Psi = A + G$ (for the prior precision matrix A defined above) is approximated by

$$\tilde{\Psi} = A + \tilde{G}, \quad (40.10)$$

where the full GGN matrix $G = J^\top H J$ (which depends on all N data points) is replaced by a low-rank approximation $\tilde{G}$ constructed from the M inducing points and the learned matrix V . This effectively substitutes the data-dependent term with a much smaller, tractable equivalent, achieving scalability while maintaining a principled variational approximation. For a detailed derivation, we refer the reader to Ortega et al. [1265].

40.2 Sampling via Iterative Solvers

The approaches discussed so far, from structured approximations like KFAC to function-space variational methods like VaLLA, all focus on constructing a tractable *approximation* to the curvature matrix Ψ or its inverse. An entirely different and powerful alternative for scalability is to sidestep the explicit approximation of Ψ altogether. Instead, this family of methods focuses on drawing samples from the posterior, $p(\theta \mid \mathcal{D}) \approx \mathcal{N}(\theta; \theta_*, \Psi^{-1})$, and then using these samples to compute posterior predictive distributions via Monte Carlo estimation. This strategy is particularly appealing because it leverages a key insight, often related to *Matheron's rule* for Gaussian conditionals [855, 1772], which has its roots in the perturb-and-optimize perspective of Papandreou and Yuille [1293] and has been operationalised as the *sample-then-optimize* (STO) principle in [36].

Definition 40.1. (*Matheron's Rule for Linear Models*). *Given a linear model with prior $p(\theta) = \mathcal{N}(0, A^{-1})$ and Gaussian likelihood $p(Y \mid \theta) = \mathcal{N}(J\theta, H^{-1})$, an exact sample from the posterior $p(\theta \mid Y) = \mathcal{N}(\theta_*, \Psi^{-1})$ can be constructed from a prior sample $\theta_0 \sim \mathcal{N}(0, A^{-1})$ and a noise sample $\epsilon \sim \mathcal{N}(0, H^{-1})$ as*

$$\theta_s = \theta_0 + A^{-1} J^\top (J A^{-1} J^\top + H^{-1})^{-1} (Y - J\theta_0 - \epsilon),$$

where J is the Jacobian, H is the likelihood precision, and A is the prior precision.

While this formula provides a theoretical basis, its direct computation still involves a large matrix inverse. A more practical application of this principle is the “sample-then-optimize” method from Antorán et al. [36], which demonstrates that an exact sample from the high-dimensional Gaussian posterior can be obtained by solving a related, but randomly perturbed, convex optimisation problem.

40.2.1 Sampling by Optimising a Perturbed Objective

Let us recall our target posterior $\mathcal{N}(\theta; \theta_*, \Psi^{-1})$, where the full-rank curvature matrix is $\Psi = G + A$. Here, G is the GGN matrix and A is the Hessian of the regulariser, or equivalently the prior precision matrix (e.g., $A = \lambda I$ for an isotropic prior). The STO method shows that a sample θ_s from the posterior can be generated by finding the minimum of a randomly perturbed version of this same objective. The procedure to draw one sample θ_s is as follows:

1. *Sample from the prior:* Draw a random parameter perturbation from the prior distribution, $\epsilon_\theta \sim \mathcal{N}(0, A^{-1})$.
2. *Sample from the likelihood:* For each data point (x_n, y_n) in the training set $\mathcal{D}$, draw a random “target” perturbation from the likelihood’s noise model, $\epsilon_{y,n} \sim \mathcal{N}(0, H_n^{-1})$, where $H_n = \nabla_z \nabla_z^\top \ell(z, y_n)|_{z=f(x_n, \theta_*)}$.
3. *Optimise the perturbed objective:* Solve for the parameter *delta* sample, $\delta\theta_s$, by minimising the following quadratic least-squares problem:

$$\delta\theta_s = \arg \min_{\delta\theta \in \mathbb{R}^P} \frac{1}{2} (\delta\theta - \epsilon_\theta)^\top A (\delta\theta - \epsilon_\theta) + \sum_{n=1}^N \frac{1}{2} (\epsilon_{y,n} - J_n \delta\theta)^\top H_n (\epsilon_{y,n} - J_n \delta\theta), \quad (40.11)$$

where J_n is the network Jacobian $J_\theta f(x_n, \theta_*)$ for data point n .

The final posterior sample is then constructed as $\theta_s = \theta_* + \delta\theta_s$. It can be shown that the solution θ_s to this procedure is an *exact sample* from the target Laplace posterior $\mathcal{N}(\theta; \theta_*, \Psi^{-1})$, and we refer to Antorán et al. [36] for additional details.

40.2.2 Iterative Solvers for Sampling

The true power of this method lies in the ability to optimise an objective instead of performing expensive matrix inversions. The objective in Equation 40.11 is a large, convex quadratic problem. Finding its minimum is equivalent to solving the linear system $\Psi\delta\theta = b$, where $b = A\epsilon_\theta + \sum_{n=1}^N J_n^\top H_n \epsilon_{y,n}$. This system can be solved efficiently with *iterative linear solvers*, avoiding any $\mathcal{O}(P^3)$ matrix inversions.

- *Conjugate Gradient (CG)*: This is a classic iterative method for solving large linear systems $\Psi x = b$ where Ψ is symmetric and positive-definite (which it is for LLA). Conjugate gradient approaches simply require matrix-vector products of the form $\Psi v = (G + A)v$, and these “GGN-vector products” can be computed efficiently using automatic differentiation on the *entire dataset* without ever forming the $P \times P$ matrix. The total cost is $\mathcal{O}(KPN)$, where K is the number of iterations needed for CG to converge, often much smaller than P ($K \ll P$).
- *Stochastic Solvers (e.g., SGD)*: For massive datasets where N is too large to process in each iteration, we can treat Equation 40.11 as a finite-sum objective and minimise it using standard stochastic optimisation. At each step, we sample a mini-batch $\mathcal{B}$ and compute a stochastic gradient of the objective. This requires a GGN-vector product *on the mini-batch* $\mathcal{B}$, and we obtain $\mathcal{O}(|\mathcal{B}|P)$ time-complexity per step. The total time is $\mathcal{O}(EPN)$ if E epochs are needed for convergence, which may be more than the full-batch CG, but it allows for trading compute time for memory.

40.2.3 Monte Carlo Predictive Distributions

Once we have a set of S posterior samples, $\{\theta_s\}_{s=1}^S$, obtained by running this procedure S times with different random perturbations, we can approximate the predictive distribution for a new input x_* using standard Monte Carlo approximations. The predictive mean of the *function* $f(x_*, \theta)$ is given by the sample mean:

$$\mathbb{E}[f(x_*, \theta) \mid \mathcal{D}] \approx \frac{1}{S} \sum_{s=1}^S f(x_*, \theta_s) =: \bar{f}(x_*). \quad (40.12)$$

The predictive covariance of the *function* is given by the sample covariance:

$$\text{Cov}[f(x_*, \theta) \mid \mathcal{D}] \approx \left(\frac{1}{S} \sum_{s=1}^S f(x_*, \theta_s) f(x_*, \theta_s)^\top \right) - \bar{f}(x_*) \bar{f}(x_*)^\top, \quad (40.13)$$

where $\bar{f}(x_*)$ denotes the sample mean from the previous equation. This covariance captures the epistemic uncertainty (model uncertainty) in the function’s output. This sampling-based approach therefore transforms the intractable problem of matrix inversion into a more manageable series of iterative optimisations, allowing us to trade off approximations for tractable computational complexity.

40.3 Model Selection in Scalable Laplace Approximations

A critical, yet challenging, component of any Bayesian method is the selection of hyperparameters. In the context of the Laplace approximation, this primarily involves the parameters of the prior distribution, $p(\theta \mid \varphi)$, which are encapsulated in the hyperparameter set φ . When we employ L2-regularisation of the form $\Omega(\theta) = \frac{1}{2} \theta^\top A \theta$, we are implicitly choosing a zero-mean Gaussian prior $p(\theta) = \mathcal{N}(0, A^{-1})$. The hyperparameters φ are thus the elements of the prior precision matrix A , which could be a single scalar precision λ (for an isotropic prior $A = \lambda I$), a set of layer-wise precisions, or a more complex structure.

40.3.1 The Marginal Likelihood Optimisation Framework

The most principled approach for setting these hyperparameters is *Type-II Maximum Likelihood*, also known as *Evidence Maximisation*. This framework treats the hyperparameters φ as parameters of a generative model for

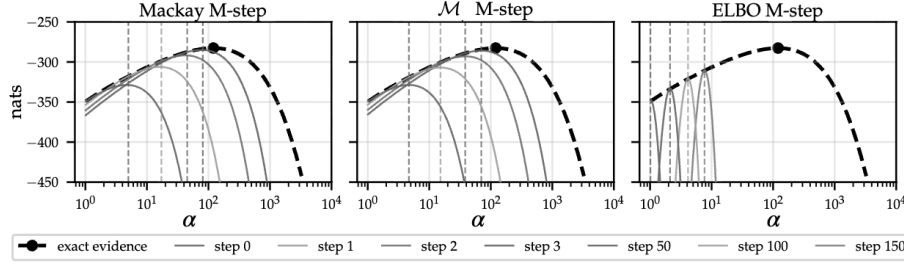


Figure 40.2: The rate of convergence of different approximate marginal likelihood algorithms compared with the exact marginal likelihood $\mathcal{M}$ (middle panel). MacKay’s first-order updates (left panel) converge almost as quickly to the optimal solution $A = \lambda_* I$, whereas the unconstrained ELBO (right panel) is very slow to converge.

the data, and seeks to find the values of φ that maximise the probability of the observed data $\mathcal{D}$. This probability is the marginal likelihood, or “evidence”:

$$\mathcal{M}(\varphi) = \log p(\mathcal{D} \mid \varphi) = \log \int p(\mathcal{D} \mid \theta) p(\theta \mid \varphi) d\theta. \quad (40.14)$$

As we established in Section 11.2, the Laplace approximation provides a closed-form estimate for this intractable integral:

$$\mathcal{M}(\varphi) \approx \underbrace{\log p(\theta_* \mid \varphi) + \log p(\mathcal{D} \mid \theta_*)}_{\text{Log-joint at MAP}} - \frac{1}{2} \log \det \Psi(\theta_*) + \frac{P}{2} \log(2\pi), \quad (40.15)$$

where θ_* is the MAP estimate (which itself depends on φ) and the curvature $\Psi = G + A$ also depends on φ . The fundamental challenge of scalability is laid bare by this equation: the log-determinant term, $\log \det \Psi$, requires $\mathcal{O}(P^3)$ computation and $\mathcal{O}(P^2)$ memory, rendering it completely intractable for modern deep neural networks. The remainder of this section is dedicated to exploring the different strategies that scalable Laplace methods employ to circumvent this computational bottleneck.

40.3.2 Optimisation via the Evidence Lower Bound (ELBO)

One general strategy for dealing with an intractable marginal likelihood is to instead optimise a tractable lower bound. Variational Inference (VI), introduced in Chapter 12, posits an approximate posterior $q(\theta)$ from a simpler, tractable family (e.g., a Gaussian) and seeks to minimise the KL divergence between this approximation and the true posterior $p(\theta \mid \mathcal{D}, \varphi)$. This minimisation is equivalent to maximising the Evidence Lower Bound (ELBO), $\mathcal{M}_{\text{ELBO}}(q, \varphi)$:

$$\log p(\mathcal{D} \mid \varphi) \geq \mathcal{M}_{\text{ELBO}}(q, \varphi) = \mathbb{E}_q[\log p(\mathcal{D} \mid \theta)] - \mathcal{D}_{\text{KL}}(q(\theta) \parallel p(\theta \mid \varphi)). \quad (40.16)$$

This objective can be jointly optimised with respect to both the variational parameters of q and the hyperparameters φ of the prior. While this avoids the log-determinant of Ψ , it introduces its own complexities. Optimising with respect to a full-covariance $q(\theta) = \mathcal{N}(\theta_q, \Sigma_q)$ requires storing and updating Σ_q , which has $\mathcal{O}(P^2)$ parameters. This often leads to further simplifications, such as assuming a diagonal covariance $\Sigma_q = \text{diag}(\sigma_q^2)$, which is a common approach in “Variational Bayes” methods but, as we saw, can be an oversimplification.

40.3.3 Optimisation via the Expectation-Maximisation (EM) Algorithm

A second major strategy for maximising the marginal likelihood (Equation 40.15) is the *Expectation-Maximisation (EM) algorithm*. In the ELBO-based approach above, the variational mean θ_q must be optimised jointly with the covariance and the prior hyperparameters, coupling the two problems. The EM algorithm proposed by MacKay [1105] (see also MacKay [1111]) decouples them, alternating between setting the variational mean equal to the exact posterior mean of θ for a given choice of A (the E-step) and updating A (the M-step), until the pair (A, θ) converges to a locally optimal setting. Starting from an initial hyperparameter estimate A_k , the algorithm iterates between two steps:

- (*E-step*) Compute the posterior over the parameters under the current hyperparameters. In our setting, this is exactly the Laplace approximation,

$$q_k(\theta) = \mathcal{N}(\theta; \theta_*, \Psi_k^{-1}) \quad \text{where} \quad \Psi_k = G + A_k. \quad (40.17)$$

- (*M-step*) Update the hyperparameters by maximising the expected log-joint under this posterior,

$$A_{k+1} = \arg \max_A \mathbb{E}_{q_k} [\log p(\mathcal{D} \mid \theta) + \log p(\theta \mid A)]. \quad (40.18)$$

While this breaks the problem down, the M-step still involves computing the next estimate of the prior precision matrix A , which can be computationally intractable, requiring us to turn to approximate, first-order solutions. Figure 40.2 illustrates the convergence behaviour of these alternatives.

MacKay’s First-Order Update

For the common case of an isotropic prior $A = \lambda I$, MacKay [1105] provided a simple and elegant first-order update rule that solves the M-step:

$$\lambda_{k+1} = \frac{\gamma_k}{\|\theta_*\|^2}. \quad (40.19)$$

Here, γ_k is the “*effective number of parameters*”, a crucial quantity defined as

$$\gamma_k = \text{Tr}(G\Psi_k^{-1}) = \text{Tr}(G(G + \lambda_k I)^{-1}). \quad (40.20)$$

This term γ_k intuitively measures the number of parameters in the model that are effectively “used” or constrained by the data, as opposed to being constrained by the prior. However, despite its elegance, this update rule is still intractable without further approximations, as calculating the trace of the matrix inverse of Ψ faces all the usual issues. Nevertheless, we mention MacKay’s first-order update here, foreshadowing a computationally tractable version of it later in the chapter.

40.3.4 Practical Hyperparameter Tuning for Scalable Methods

We can now see how each of the scalable methods discussed in this chapter navigates these challenges.

Structured Approximations (Diagonal and KFAC)

These methods take the most direct route. They replace the intractable curvature Ψ with a structured approximation $\tilde{\Psi}$ (diagonal or KFAC) that is explicitly designed to be tractable. The core benefit is that $\log \det \tilde{\Psi}$ is cheap to compute. For a diagonal $\tilde{\Psi}$, it is $\mathcal{O}(P)$. For KFAC, it scales with the sum of cubes of layer dimensions, not $\mathcal{O}(P^3)$. These methods can therefore perform Type-II Maximum Likelihood by directly optimising the approximate marginal likelihood. This is typically done using gradient-based methods, computing $\nabla_{\varphi} \mathcal{M}(\varphi)$ and taking gradient steps on the prior hyperparameters (e.g., a single λ or layer-wise λ_i).

Subnetwork and Last-Layer Inference

These methods face a more fundamental problem. Partitioning the parameters into a Bayesian set θ_B and a deterministic set θ_D implicitly places a Dirac-delta prior on θ_D . A Dirac delta is not absolutely continuous with respect to the Lebesgue measure and therefore admits no density; evaluating the log-marginal likelihood (Equation 40.15) requires treating $\delta(\cdot)$ as a density, whose value diverges at the point mass, and any KL divergence between the continuous Gaussian component and the degenerate delta component is infinite. The failure is thus a measure-theoretic mismatch—a lack of absolute continuity—rather than merely “disjoint support”. Lacking a principled Bayesian objective, these methods must fall back on the standard machine learning practice of *cross-validation*. The prior precision λ is treated as a regularisation hyperparameter to be tuned by minimising a score (e.g., negative log-likelihood) on a held-out validation set. For subnetwork inference, Daxberger et al. [381] also propose a heuristic to adjust the prior: if a full network of size P uses precision λ , a subnetwork of size P_S should use $\lambda_S = \lambda \cdot (P_S/P)$. This attempts to keep the function-space variance (the magnitude of the implied GP kernel) approximately constant.

Variational LLA (VaLLA)

Although VaLLA is formulated as a sparse variational GP, it does not optimise the standard ELBO. This is not because a fixed mean function is inherently incompatible with the ELBO—decoupled sparse GPs with fixed or pre-trained means can be trained with the standard variational objective—but because of how the fixed mean interacts with the non-conjugate likelihood once only the covariance is learned. With the mean anchored to $f(x, \theta_*)$, the

zero-forcing character of the KL term drives the posterior variances towards a degenerate collapse, systematically underestimating predictive uncertainty. To avoid this, VaLLA employs an alternative variational objective based on the α -divergence for $\alpha \in (0, 1]$:

$$\max_{q(f)} \sum_{i=1}^N \frac{1}{\alpha} \log \mathbb{E}_q [p(y_i | f)^\alpha] - \mathcal{D}_{\text{KL}}(q \| p). \quad (40.21)$$

Compared with the ELBO, the objective in Equation 40.21 is mass-covering rather than zero-forcing: it penalises underestimated variances more heavily, preventing the collapse and enabling a stable, principled optimisation of the prior hyperparameters.

Sampling-Based EM Algorithm

Finally, the sampling-based approach from Section 40.2 provides a powerful way to implement the EM algorithm in a fully scalable manner. We want to use MacKay’s update (Equation 40.19), but we must compute the intractable trace $\gamma_k = \text{Tr}(G\Psi_k^{-1})$. We can thankfully rely on *stochastic trace estimation* as a solution. The trace of a matrix Λ can be written as an expectation $\text{Tr}(\Lambda) = \mathbb{E}_{v \sim p(v)}[v^\top \Lambda v]$, where $p(v)$ is a distribution with zero mean and identity covariance. In our case, this can be cleverly rearranged to be an expectation over zero-mean posterior samples:

$$\gamma_k = \text{Tr}(G\Psi_k^{-1}) = \mathbb{E}_{\bar{\theta} \sim \mathcal{N}(0, \Psi_k^{-1})} [\bar{\theta}^\top G \bar{\theta}]. \quad (40.22)$$

The identity in Equation 40.22 can be approximated with a Monte Carlo average. We can now combine all the pieces into a fully scalable algorithm:

1. Start with an initial hyperparameter λ_k .
2. (*E-step, implicitly*) Draw S zero-mean samples $\{\bar{\theta}_s\}_{s=1}^S$ from $\mathcal{N}(0, \Psi_k^{-1}) = \mathcal{N}(0, (G + \lambda_k I)^{-1})$ using the *sample-then-optimize (STO)* procedure from Section 40.2, which relies only on iterative linear solvers. Note that the minimiser of Equation 40.11 is the parameter *delta* $\delta\theta_s \sim \mathcal{N}(0, \Psi_k^{-1})$, which is exactly the zero-mean sample required here: we set $\bar{\theta}_s \equiv \delta\theta_s$ and do *not* add the MAP centre θ_* . Using the shifted posterior samples $\theta_s = \theta_* + \delta\theta_s$ instead would bias the trace estimator by $\theta_*^\top G \theta_*$.
3. (*M-step, stochastically*) For each sample $\bar{\theta}_s$, compute the GGN-vector product $v_s = G\bar{\theta}_s$. This is computationally cheap, with the same cost as the iterative solver steps. We then estimate the trace, $\hat{\gamma}_k = \frac{1}{S} \sum_{s=1}^S \bar{\theta}_s^\top v_s$, and finally apply MacKay’s update: $\lambda_{k+1} = \hat{\gamma}_k / \|\theta_*\|^2$.
4. Repeat until convergence.

This sampling-based EM algorithm elegantly sidesteps the $\mathcal{O}(P^3)$ bottleneck, relying only on efficient matrix-vector products.

40.4 Concluding Remarks

In this chapter, we have explored a wide variety of techniques to scale the highly flexible Laplace approximation to very large neural networks on very large datasets. As we saw, there are multiple approaches to tackle the problem, ranging from approximations to the curvature matrix, to exploiting the function-space duality with sparse GPs, and finally to simplifying the task into stochastic estimates that can be approximated with samples from the posterior. Across all these methods, there are various trade-offs being made between time and space complexity, and the flexibility of the approximation being chosen. Depending on the problem at hand, the Laplace approximation is amenable to a veritable toolbox of scalable approaches for the practitioner.

Chapter 41

Learned Compression with Deterministic Codes

The ever-increasing amount of information that is being generated and transmitted globally demands increasingly powerful data compression algorithms. Although traditionally studied under information theory [1499] and signal processing [611], data compression is inherently a problem of probabilistic modelling and a fertile ground for the application of Bayesian deep learning methods. At its core, compression works by exploiting the structure and patterns inherent in the data, so that more predictable patterns can be described with fewer bits. It is then no surprise that statistical learning, which aims to understand structure in data, is built into such celebrated compression algorithms as adaptive arithmetic coding [1400] and Lempel-Ziv [1884] (the basis of extremely popular file formats such as `zip` and `gzip`). In the reverse direction, compression ideas have also profoundly influenced machine learning, starting from Occam’s razor, to the principles of Minimum Description Length [676] and Information Bottleneck [1613, 1614], and to the current Large Language Models whose roots trace back to Shannon’s landmark treatise on information theory [1499].

In recent years, the development of sophisticated statistical models (i.e., “deep generative models”) such as variational autoencoders (VAEs) [892], generative adversarial networks (GANs) [650], and normalizing flows [1290] is having an increasing impact on data compression. Information theory tells us that any probability model for discrete data corresponds to a lossless compression algorithm [924, 1160], and various approaches have been developed for neural lossless compression based on (variational) autoencoders [1167, 1625], normalizing flows [754, 778, 1630], autoregressive models [779], and diffusion models [893]. Deep generative models have similarly reshaped lossy compression. Ideas for neural lossy compression can already be found in early work on hierarchical latent variable models [664], and were subsequently made competitive in rate-distortion performance by end-to-end training with an effective entropy model and approximation to quantization [83, 1596]. Subsequent methods further incorporate GANs [10, 1168] or diffusion models [624, 781, 1597, 1801, 1809] for enhanced realism or perceptual quality, allowing highly realistic reconstructions to be generated at extremely low bit-rates.

In this and the next chapter, we will introduce elements of information theory and the basics of data compression, and explore how ideas from deep probabilistic modelling are closely related to compression and can be used to build powerful compressors.

We start by considering lossless compression (Section 41.2), where the goal is to assign data points to strings of bits in a one-to-one manner, as efficiently as possible. A basic result in information theory guarantees that any probability model induces a valid assignment and vice versa. Therefore, we can (at least in principle) reduce the problem of lossless compression to that of estimating the data distribution. To simplify modelling high-dimensional discrete distributions, we consider either factoring our model distribution autoregressively, or formulating it as a latent variable model with a tractable joint distribution. Each modelling choice requires a different compression procedure. In the first case, an autoregressive model lends itself naturally to compression using an algorithm called *arithmetic coding* (Section 41.2.5). In the second case, we first infer the latent representation of a given data point, and then compress them both using their joint distribution under the model (Section 41.2.5). The resulting “two-part code” turns out to have a suboptimal code length, which is addressed by bits-back coding (Section 42.1, next chapter). In lossy compression (Section 41.3), a data point is first encoded into a compressed latent representation, from which a reconstruction is then later decoded. As we allow imperfect reconstructions, a trade-off must be made between the amount of acceptable *distortion* in the lossy reconstruction and the (bit-) *rate* of the compressed representation (as captured by the file size). Inspired by vector quantization and transform coding methods (Section 41.3.1), we build neural lossy compressors based on *nonlinear transform coding* (Section 41.3.2)

that are optimized end-to-end for rate-distortion performance. As we elaborate in Section 41.3.3, there is a strong parallel between a lossy compressor and a latent variable model, where the encoding procedure corresponds to variational inference and the decoding procedure corresponds to conditional data generation. This parallel can be made precise in the case of nonlinear transform coding, and later with the help of *Relative Entropy Coding* (Section 42.2, next chapter), which promises to turn any latent variable model into a lossy compressor.

In both lossless and lossy compression, a latent variable model has a role to play, and we can draw connections and inspirations from variational inference and learning. In lossless compression, the latent variables help define a probability model over the data space via marginalization, and the negative evidence lower bound (ELBO, Chapter 12), or “NELBO”, represents an upper bound on the ideal coding cost under the marginal probability model (see Section 42.1.4). In lossy compression, the latent variables often represent a decorrelated representation of the data that can be more effectively quantized then compressed, and the NELBO represents a rate-distortion cost (see Section 41.3.3). In both cases, (approximate) variational inference produces the optimal encoding for a given latent variable model; training the model on the ELBO with amortized inference then corresponds to learning a data compressor end-to-end.

Although learning-based methods have consistently demonstrated strong compression performance in literature, building and deploying compression systems in practice faces many engineering challenges such as numerical issues [86] and computational efficiency [847, 1804]. We sweep these issues under the rug and focus on the conceptual foundations. Although we discuss learned compression in the data compression setting, the same techniques can also be adapted to compress the weights of neural networks in a straightforward manner [1256, 1806]. We refer readers to libraries such as TensorFlow Compression [87] and CompressAI [108] which provide open-source implementations of learning-based compression methods and serve as a convenient starting point for their own research.

41.1 What is Compression?

Data compression aims to reduce the number of bits needed to represent useful information in the data. This raises such basic questions as “*What is information?*” and “*What does it mean to communicate information?*”. We find answers in probability theory, following the approach of Claude Shannon who first studied these questions and founded the field of information theory [1499]. We view the information-producing *source*, i.e., the data to be compressed, as a random object X , which takes values in a predetermined set $\mathcal{X}$ known as the *alphabet*. To compress data, the sender observes some value of $X = x$ and *encodes* it into a short sequence of bits; this is done in such a way that the receiver, after retrieving the same sequence of bits via a perfect binary channel, can then *decode* it into a data reconstruction. For example, in image compression, the data source may be an N -dimensional digital image X that takes value in the space of pixels $\mathcal{X} = \{0, 1, \dots, 255\}^N$. The sender observes an image x and uses a compression algorithm (such as JPEG) to encode it into a binary file; the same binary file can then be transmitted to a receiver over a digital channel such as the Internet, and decoded back into a reconstruction of the same image.

Thus we see that a basic problem of compression is that of assigning bit-strings to data observations. If the data can always be perfectly reconstructed under the assignment scheme, we say the compression is *lossless*, and otherwise *lossy*. Since the set of bit-strings is countable, lossless compression is only possible when the data is discrete. As we shall see, many lossy compression methods build on lossless compression as a subroutine; this is done by first transforming the data into a discrete representation, which can then be losslessly compressed/received by the receiver and finally transformed into a reconstruction. This chapter will introduce the fundamentals of data compression and its relation to statistical learning, resulting in learned (or neural) compression methods. Most commonly, the assignment from each data observation to a bit-string is a *deterministic* function, often facilitated by quantization. Chapter 42 will instead discuss more advanced methods which use *stochastic* instead of deterministic encoding.

41.2 Lossless compression

Lossless compression aims to represent discrete data with as few bits as possible, while being able to perfectly recover the data later. Formally, we consider the general problem where the source consists of i.i.d. copies of a discrete random variable X taking values in $\mathcal{X}$. To each possible *message* $(x_1, x_2, \dots) \in \bigcup_{n=1}^{\infty} \mathcal{X}^n$ (i.e., a realized source sequence), our goal is to assign a unique string of bits (which we call a *bit-string*) while minimizing the overall number of bits required. Such an assignment of messages to bit-strings that allows us to perfectly recover the former from the latter is known as a lossless *source code*. Since the set of all possible bit-strings $\{0, 1\}^* := \{\emptyset, 0, 1, 00, 01, \dots\}$

is countable and we require a unique bit-string for every possible message, lossless compression is only possible for a discrete source. The structure of $\mathcal{X}$ is not important for our conceptual discussions, and we can, without loss of generality, identify $\mathcal{X}$ with the set of natural numbers. Still, a reduction in the number of bits is never guaranteed, and lossless compression relies on the fact that certain outcomes of the source are more probable than others and can be assigned shorter bit-strings. Therefore, we cannot expect to compress a “pure noise” source whose outcomes are all equiprobable. Fortunately, real-world sources (such as the English language) often contain structure that makes a small subset of outcomes much more likely than others (e.g., the string “Hello world!” vs. “bwzjnhxiHcBuq”).

To quantify the amount of uncertainty in a source, Shannon [1499] made the following definition: for a discrete random variable X with probability mass function (PMF) P_X , its *entropy* is

$$H[X] := \mathbb{E} \left[\log_2 \frac{1}{P_X(X)} \right] \quad (41.1)$$

$$= \sum_{x \in \mathcal{X}} P_X(x) \log_2 \frac{1}{P_X(x)}. \quad (41.2)$$

For each outcome $x \in \mathcal{X}$, the number $\log_2 \frac{1}{P_X(x)}$ is known as *self-information* and quantifies the amount of surprise of observing $X = x$; entropy is therefore the expected value of surprise. For example, the entropy of a fair coin flip — a Bernoulli(0.5) random variable — is 1 bit, and the entropy of a biased coin flip approaches 0 bits as the probability of one of the two outcomes approaches 1. The definition of entropy can be motivated by natural axioms of “information content” [1499] and turns out to delineate the fundamental performance limit of lossless compression. In the rest of this section, we will give a mathematical description of this performance limit and introduce algorithms that aim to approach this limit, known as *entropy coding* algorithms.

41.2.1 Optimal Symbol Code

We first consider codes that assign a bit-string to one letter of the message at a time, i.e., we want to find a *symbol code* $\gamma : \mathcal{X} \rightarrow \{0, 1\}^*$, and obtain the bit-string for an entire message by concatenation. We call each assigned bit-string $\gamma(x)$ for the symbol $x \in \mathcal{X}$ a *codeword*; we call the number of bits making up $\gamma(x)$ the *code length*, variously denoted by $l(x)$ or $|\gamma(x)|$. For the assignment to be lossless, clearly γ needs to assign a unique codeword to each x , so that the receiver can recover x unambiguously from $\gamma^{-1}(\cdot)$. To allow the efficient decoding of a message from a sequence of concatenated codewords, we impose the further condition that γ is a *prefix-free code*, i.e., no codeword under γ is a prefix of another.

It turns out that any prefix-free code with code lengths l must satisfy the following *Kraft inequality*,

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1, \quad (41.3)$$

and conversely, any code-length function l satisfying the above inequality can always be realized by some prefix-free code [337, Theorem 5.2.1].¹ As a consequence, the optimal average code length among all prefix-free (or uniquely decodable) codes is given by the following optimization problem:

$$\begin{aligned} L^*(X) &:= \min_{\gamma} \sum_{x \in \mathcal{X}} P_X(x) l(x), \\ \text{s.t. } &\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1, \\ &l(x) \in \mathbb{N} \quad \forall x \in \mathcal{X} \end{aligned}$$

By studying this problem, we arrive at the following basic result which bounds the optimal average code length of prefix-free codes in terms of entropy [337, Theorem 5.4.1]:

$$H[X] \leq L^*(X) \leq H[X] + 1 \quad \text{bits per source sample.} \quad (41.4)$$

The first inequality states that no prefix-free code (or more generally, no *uniquely decodable* code) can achieve an average code length below the source entropy, and the inequality is tight when $l(x) = \log_2 \frac{1}{P_X(x)}$. The second

¹The result also holds if we replace “prefix-free code” with “uniquely decodable code” (see [337, Chapter 5.5]). Although the former is a subset of the latter, it turns out we do not lose any coding efficiency by considering prefix-free codes only.

inequality says that there always exists a prefix-free code whose average code length is one bit within the entropy; the existence is guaranteed since the code lengths $l(x) = \lceil \log_2 \frac{1}{P_X(x)} \rceil$ (we use the ceiling operator $\lceil \cdot \rceil$ to ensure integer code lengths) satisfy the Kraft inequality. This result has a significant implication: given any discrete distribution Q , an optimal code for $X \sim Q$ always exists whose code lengths are approximately negative log-probabilities $\log_2 \frac{1}{Q(x)}$; we will give an explicit construction in Section 41.2.3. Thus, the problem of finding an optimal lossless compressor reduces to that of finding an optimal coding distribution, which we turn to next.

41.2.2 Lossless Coding and Statistical Estimation

In practice, the true distribution P_X of the source is almost always unknown. We therefore approximate it by a model P_θ , called an *entropy model* or *coding distribution*, and proceed with entropy coding as if the data came from the distribution P_θ . Using a code designed for P_θ with code length $l(x) = \log_2 \frac{1}{P_\theta(x)}$, the expected code length then satisfies [337, Theorem 5.4.3]

$$H[P_X, P_\theta] \leq \mathbb{E}_{x \sim P_X} [l(x)] \leq H[P_X, P_\theta] + 1, \quad (41.5)$$

where

$$H[P_X, P_\theta] \stackrel{\text{def}}{=} \mathbb{E}_{x \sim P_X} \left[\log_2 \frac{1}{P_\theta(x)} \right], \quad (41.6)$$

is the *cross-entropy* between P_X and P_θ and is related to entropy via

$$H[P_X, P_\theta] = H[X] + D_{\text{KL}}(P_X \| P_\theta) \geq H[X]. \quad (41.7)$$

Above, the Kullback-Leibler (KL) divergence $D_{\text{KL}}(P_X \| P_\theta) \stackrel{\text{def}}{=} \mathbb{E}_{x \sim P_X} \left[\log_2 \frac{P_X(x)}{P_\theta(x)} \right]$ is always positive unless $P_X = P_\theta$. Thus, coding $X \sim P_X$ using an incorrect coding distribution P_θ incurs an increase in the average description length given by $D_{\text{KL}}(P_X \| P_\theta)$.

Consequently, fitting a model P_θ by maximum-likelihood estimation, i.e.,

$$\arg \max_{P_\theta} \mathbb{E}_{x \sim P_X} [\log_2 P_\theta(x)] = \arg \min_{P_\theta} \mathbb{E}_{x \sim P_X} [-\log_2 P_\theta(x)], \quad (41.8)$$

is precisely aligned with minimizing the coding cost of data from P_X under an imperfect model P_θ . Note that in practice we only have access to samples from the data distribution P_X , so that the above expectations are approximated by Monte Carlo averages and often optimized using stochastic gradient descent [160].

41.2.3 Huffman Coding

As an example, we review one of the most widely used entropy coding algorithms due to Huffman [803], which achieves the optimal prefix-free code length in (41.4). In a nutshell, given a probability mass function (PMF) P_X over symbols, the algorithm first derives an optimal prefix-free code from a so-called Huffman tree; any symbol can then be encoded and decoded by traversing the tree. Figure 41.1 shows an example Huffman tree, whose leaf nodes are associated with symbols of the alphabet. A Huffman tree is constructed recursively from the leaves to the root by successively merging the two nodes with the lowest probabilities into a new node. By traversing the tree from the root to a leaf, the sequence of branching directions (“0” for left, “1” for right, in our example) assigns a unique binary codeword to each symbol, with the more frequent symbols receiving shorter codewords. Encoding and decoding are simple lookup operations with complexity linear in the tree depth.

Huffman coding assigns a codeword with length at most $\lceil -\log_2 P_X(x) \rceil$ to each symbol x , and can be shown to be optimal among all prefix-free codes [337]. However, as log-probabilities are generally not integers, Huffman coding (as with all so-called *symbol codes*) incurs an overhead of up to 1 bit per symbol, relative to the information content. For example, a heavily biased coin with entropy close to 0 will still require 1 bit to encode.

One way of reducing this overhead of symbol codes is by spreading the overhead over a group of symbols.² Given a message $(x_1, x_2, \dots)$, we consider transmitting it in length- n blocks with a symbol code, effectively treating the source as i.i.d. length- n blocks with a new alphabet $\mathcal{X}^n$. The bound on the optimal prefix-free code length (Eq. (41.4)), averaged over each symbol, then becomes

$$\frac{1}{n} H[X_1, \dots, X_n] \leq \frac{1}{n} L^*([X_1, \dots, X_n]) \leq \frac{1}{n} H[X_1, \dots, X_n] + \frac{1}{n}. \quad (41.9)$$

²Indeed, Shannon [1499, Theorem 9] already used block coding and the law of large numbers to show that an i.i.d. source can be losslessly compressed with an average code length arbitrarily close to the entropy. However, it was not until decades later with the invention of arithmetic coding that an algorithm made this idea a practical reality.

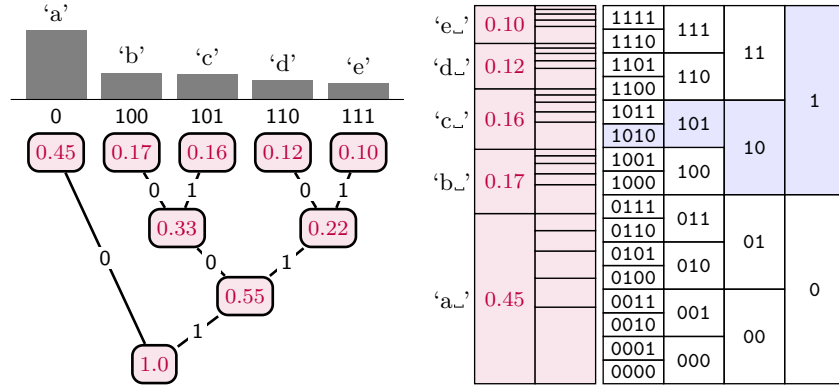


Figure 41.1: *Left*: A Huffman tree over five symbols. The numbers in boxes denote probabilities, while binary strings above the leaf nodes correspond to the codewords assigned by the Huffman algorithm. *Right*: Arithmetic coding of a message of length two. Each message corresponds to an interval whose length is proportional to the message's probability. As an example, any real number inside the interval corresponding to the message 'ca' begins with 0.1010... when written in binary form so that we can use 1010 to uniquely encode it. The message 'aa' would be encoded as 000.

Thus, the 1 bit overhead on the RHS of (41.4) is reduced to $\frac{1}{n}$ and can be made arbitrarily low by increasing n . Note that in the usual case where $X_1, X_2, \dots$ are i.i.d., the joint entropy factorizes so that $\frac{1}{n}H[X_1, \dots, X_n] = H[X]$; otherwise, this block coding scheme can exploit the redundancy between the X_i s and do better than coding one symbol at a time, since

$$H[X_1, \dots, X_n] \leq \sum_{j=1}^n H[X_j],$$

for any n random objects $(X_1, \dots, X_n)$.³

Despite the theoretically improved coding cost, directly applying a symbol code like Huffman code over long blocks is often computationally infeasible, as the size of the required PMF table for $P_{X_1, \dots, X_n}$ grows exponentially with the block length. It turns out that *stream codes* [1111, Chapter 6] to be discussed below will allow us to implement the idea behind block coding much more efficiently, while also offering a convenient mechanism for adapting our model to the source (we discussed modelling in Section 41.2.2) *during* compression.

41.2.4 Arithmetic Coding

Arithmetic coding [1400], also known as range coding [1137], is an example of a *stream code* [1111, Chapter 6]. Stream codes differ from symbol codes in that they assign codewords to entire messages and individual symbols do not have unique codewords. Stream codes amortize the coding cost's overhead across the whole sequence of symbols and are therefore able to get closer to the entropy.

The basic idea of arithmetic coding is to associate each symbol with a subinterval of the interval $[0, 1]$ such that the subinterval's length equals the symbol's probability. This procedure can be generalized towards encoding symbols in sequence. For two symbols, the second symbol is no longer considered a subinterval of $[0, 1]$ but a subinterval of the previous symbol's interval such that the second subinterval's length equals the two symbols' joint probability. For example, if symbol "a" has probability $1/5$ and is associated with $[0, 0.2)$, the sequence "aa" would be encoded as $[0, 0.04)$. This procedure can be iterated for sequences of arbitrary length, leading to a sequence of subintervals of decreasing size that contain each other. Any real-valued number contained in the final subinterval allows us to uniquely reconstruct the sequence of symbols and therefore encodes the entire sequence (Figure 41.1). In practice, one picks a representative number which can be represented with the smallest number of bits while uniquely identifying the interval. By construction, the size of the final sub-interval is the product of all previous symbols' conditional probabilities, hence the probability of the sequence. Intuitively, the interval's length (i.e., the sequence's joint probability) determines the number of relevant digits of the number representing the interval, thus the code length equals the log-probability of the sequence as required for entropy codes. If $x^n = (x_1, x_2, \dots, x_n)$ is a message of length n , and $\gamma(x^n)$ the codeword assigned by arithmetic coding using a

³In information theory, the most general representation of a source is a stochastic process $(X)_{i=1}^\infty$, and $\lim_{n \rightarrow \infty} \frac{1}{n}H[X_1, \dots, X_n]$ defines the *entropy rate* of the process, which generalizes entropy (41.2) to non-i.i.d. sources.

probability model P_θ , then the assigned code length satisfies [1340, Chapter 13.1],

$$-\log_2 P_\theta(x^n) \leq |\gamma(x^n)| < \lceil -\log_2 P_\theta(x^n) \rceil + 1.$$

Thus if the true distribution of messages is P_{X^n} , then the average code length is at most $H[P_{X^n}, P_\theta] + 2$. The excess bits needed for encoding the message are amortized over all n symbols, making arithmetic coding more efficient on long messages than symbol codes. Per symbol, the overhead is only $2/n$ bits, compared to 1 bit per symbol for Huffman coding.

In arithmetic coding, the probability model $P_\theta(x^n)$ is only specified sequentially via the marginal $P_\theta(x_1)$ and conditional distributions $\{P_\theta(x_i|x_{<i}) : i = 2, \dots, n\}$, thereby avoiding the exponential complexity of block coding (see discussion in the previous subsection). Moreover, a simple modification allows arithmetic coding to adapt its probability model “on the fly” on the data being compressed. The idea is as follows. Before processing an input message x^n , both the sender and receiver start with the same copy of the marginal model $P_\theta(x_1)$. At each step i , the sender updates its model distribution $P_\theta(x_i|x_{<i})$ based on the encoded sequence $x_{<i}$ so far, and then encode x_i under the updated model; this is repeated for $i = 2, \dots, n$, until a final bit-string is produced. The receiver then decodes the message from the bit-string as usual, except the exact same sequence of model updates are also performed based on the decoded sequence $x_{<i}$ at each time step $i = 2, \dots, n$; this is to ensure the same model $P_\theta(x_i|x_{<i})$ is used during both encoding and decoding. Adaptive arithmetic coding via sequential density estimation is thus an example of a *universal* source code [337, Chapter 13], in the sense that the method can be applied without prior knowledge of the source distribution while still being asymptotically optimal for a large class of sources (e.g., all finite-order Markov chains).

Notably, arithmetic coding implements a first-in-first-out data structure, that is, a queue. Symbols which are encoded first are also decoded first. As we will see next, this makes it a convenient choice for compression with autoregressive models.

41.2.5 Neural Lossless Compression

As we saw in Section 41.2.3, an entropy code assigns a sequence of bits to each possible value x in the alphabet based on the PMF P_X , so the size of the alphabet (i.e., the set of all possible symbols) needs to be manageable. When working with high-dimensional sources such as images, naively representing the data distribution as a PMF table becomes infeasible, as the number of table entries increases exponentially with the dimensionality. Additionally, we would like our model P_θ to be as close as possible to the true distribution P_X (in KL divergence), so as to achieve the optimal lossless compression cost (see Eq. (41.8)). Fortunately, deep likelihood-based models parametrised by neural networks, such as autoregressive models [115, 975, 1655] and variational autoencoders [892, 1391], provide highly flexible and accurate models of the data distribution without working with the intractable PMF table. By decomposing a high-dimensional distribution into lower-dimensional conditional distributions, such likelihood-based models also provide means for tractable entropy coding using existing entropy codes such as arithmetic coding. Thus neural lossless compression, i.e., lossless compression with a neural probability model, generally follows a two-step recipe:

1. Estimate a probability model P_θ from samples of the true data distribution P_X by maximizing likelihood (Eq. (41.8)) or a surrogate criterion.
2. Compress or decompress any new data observation x with an entropy coding algorithm (e.g., arithmetic coding), using P_θ as the entropy model.

Below we instantiate this recipe with two choices of models: autoregressive models and latent-variable models.

Neural Lossless Compression with Autoregressive Models

An autoregressive model P_θ exploits the fact that we can write any probability distribution as a product of conditional distributions using the chain rule of probabilities [160],

$$P_\theta(x) = \prod_i P_\theta(x_i | x_{<i}). \quad (41.10)$$

Here, x_i is the i -th entry of the vector x and $x_{<i}$ corresponds to all previous entries in an arbitrary order. The autoregressive factorization does not make any assumptions about the data distribution yet still allows us to easily incorporate useful assumptions. For example, a Markov assumption can be implemented by conditioning only on a subset of entries in $x_{<i}$ that are closest to i . A stationarity assumption is easily incorporated by using the

same conditional distribution $P_\theta(x_i | x_{<i})$ at every location. These two assumptions are often reasonable and can drastically reduce the amount of parameters of a model.

Autoregressive modelling lends itself naturally to lossless compression in combination with arithmetic coding (Section 41.2.4) since both deal with data sequentially. Given a pretrained model P_θ , we can encode (decode) a given data vector x one symbol at a time, where at each step we encode (decode) x_i given the previous $x_{<i}$ using arithmetic coding, feeding it the model's conditional distribution $P_\theta(x_i | x_{<i})$. Arithmetic coding is particularly suited for this task as it operates like a queue — symbols encoded earlier will also be decoded earlier, and therefore will be available as input to the model for computing conditional distributions.

Autoregressive models have achieved strong likelihoods for various data modalities such as images, audio, and text, making them powerful lossless data compressors in principle. However, they come with the restriction that decoding (as with sampling) is an inherently sequential procedure: each symbol can only be decoded after all the symbols have been decoded on which its prediction depends. This can make them impractically slow at compressing high-dimensional signals such as images, and recent research [779, 1385] has sought to address this limitation.

Neural Lossless Compression with Latent Variable Models

A latent-variable model represents the data distribution using the sum rule of probability,

$$P_\theta(x) = \sum_{z \in \mathcal{Z}} P_{X|Z=z}^\theta(x) P_Z^\theta(z),$$

where $\mathcal{Z}$ is a latent space, P_Z^θ is a distribution over $\mathcal{Z}$, $P_{X|Z}^\theta$ is a conditional distribution (i.e., $P_{X|Z=z}^\theta$ is a distribution over $\mathcal{X}$ for each value of z), and we refer to P_Z^θ and $P_{X|Z}^\theta$ as the *prior* and (*conditional*) *likelihood* models in analogy to Bayesian statistics. Unlike in an autoregressive model, evaluating the model probability here is not so straightforward as it involves an expensive marginalization over the latent space. We first focus on the more basic question of how a given data point x can be entropy-coded efficiently with a code length of $\approx -\log_2 P_\theta(x)$, i.e., its self-information, assuming a pretrained latent variable model. The coding procedure will then suggest a training procedure that allows us to learn an effective neural compressor from data.

As a first attempt, we will implement lossless compression using a so-called *two-part code* [676]. Given x , the sender first decides on a latent representation z^* computed with an encoding function $f : \mathcal{X} \rightarrow \mathcal{Z}$; then z^* is encoded under the prior P_Z^θ , and x is encoded under the likelihood model $P_{X|Z=z^*}^\theta$. Upon receiving the encoded bit-streams, the receiver decodes z^* and then x using their copy of the same models. The overall code length is therefore $-\log P_{X|Z=z^*}^\theta(x) - \log P_Z^\theta(z^*)$, and we can optimize this compression scheme on the expected code length,

$$\mathbb{E}_{x \sim P_X, z=f(x)} [-\log P_{X|Z=z}^\theta(x) - \log P_Z^\theta(z)], \quad (41.11)$$

with respect to our choices of encoder f , prior P_Z^θ , and likelihood $P_{X|Z}^\theta$ models. Two comments are in order. First, unlike in standard Bayesian statistics, here we *learn* the prior P_Z^θ so as to optimize the compression performance. The relevant term in the objective is $\mathbb{E}_{x \sim P_X, z=f(x)} [-\log P_Z^\theta(z)]$, which is equal to the cross-entropy $H[P_Z, P_Z^\theta]$ where $P_Z(k) \stackrel{\text{def}}{=} \mathbb{P}(f(X) = k) = \sum_{x: f(x)=k} P_X(x)$ is the marginal distribution of the encoding $Z \stackrel{\text{def}}{=} f(X)$ and equals the pushforward of P_X under f . Thus the optimal prior is P_Z , as we saw in Section 41.2.2. Second, to allow for entropy coding, the latent space $\mathcal{Z}$ must be discrete. If we represent f with a neural network or some other smooth function approximator ψ , then we must *quantize* its output to obtain a discrete latent representation, i.e., $z = \llbracket \psi(x) \rrbracket$, where $\llbracket \cdot \rrbracket$ denotes quantization, e.g., element-wise rounding to a predefined grid. As quantization has zero gradient almost everywhere, a differentiable approximation is typically used for gradient-based training, which we discuss in more depth in Section 41.3.2. As an example, Mentzer et al. [1167] train a two-part-code model $(\psi, P_Z^\theta, P_{X|Z}^\theta)$ end-to-end on Eq. (41.11) for image compression. The model resembles a hierarchical autoencoder, in which ψ is a downsampling convolutional neural network, $z = \llbracket \psi(x) \rrbracket$ is a collection of convolutional feature tensors at decreasing spatial resolutions, and $P_Z^\theta(z) = \prod_l P^\theta(z_l | z_{<l})$ and $P_{X|Z}^\theta$ are autoregressive models parametrised by an upsampling convolutional neural network. Notably, each of the conditional distributions that make up P_Z^θ or $P_{X|Z}^\theta$ is fully factorized across pixel locations, resulting in orders of magnitude faster decoding than pixel-space autoregressive models like PixelCNN++ [1452].

However, even when optimized end-to-end, the coding cost of a two-part code is generally suboptimal [561, 776].

To see this, fix a data point x and consider the following upper bound on the self-information,

$$\begin{aligned} -\log P_\theta(x) &\leq -\log P_\theta(x) + \text{D}_{\text{KL}}\left(Q_{Z|X=x} \parallel P_{Z|X=x}^\theta\right) \\ &= \mathbb{E}_{z \sim Q_{Z|X=x}} [-\log P_Z^\theta(z) - \log P_{X|Z=z}^\theta(x) + \log Q_{Z|X=x}(z)], \end{aligned} \quad (41.12)$$

where $Q_{Z|X=x}$ is any *variational posterior* distribution, $P_{Z|X=x}^\theta(z) \propto P_Z^\theta(z)P_{X|Z=z}^\theta(x)$ is the Bayesian posterior, and Eq. (41.12) is known as the negative *ELBO* (Evidence Lower BOund) [170]. The two-part code corresponds to the special case where $Q_{Z|X=x} = \delta_{z^*}(x)$ is a degenerate distribution concentrated on the encoding $z^* = f(x)$, and the negative ELBO exactly equals the two-part code length $-\log P_{X|Z=z^*}^\theta(x) - \log P_Z^\theta(z^*)$. Thus, for a given choice of prior and likelihood models, we can in principle improve the coding cost of a two-part code by allowing ourselves the possibility of a *stochastic encoder* $Q_{Z|X}$, ideally the Bayesian posterior. This prompts the question “does there exist a compression scheme that can make use of a stochastic encoder $Q_{Z|X}$ and achieve a coding cost equal to the negative ELBO?” If so, we can then reduce the problem of optimal compression to that of posterior inference (Chapter 12), and the problem of learning such a compressor from data samples to that of maximizing the expected ELBO. Fortunately the answer is yes, and is given by the *bits-back coding* algorithm [560, 751] which we explain in Section 42.1.

41.3 Lossy Compression

In lossy compression, information in the data is irrevocably discarded, allowing us to trade away perfect reconstruction for a lower bit-rate than what is possible with lossless compression. Mathematically, a lossy compression procedure can be described by the following Markov chain

$$X \rightarrow Z \rightarrow \hat{X}.$$

Here, the source random variable $X \in \mathcal{X}$ is only available to the sender, who decides on a compressed *representation* $Z \in \mathcal{Z}$ from a representation space $\mathcal{Z}$ and transmits it to the receiver. The receiver then produces a *reconstruction* $\hat{X} \in \hat{\mathcal{X}}$ on the basis of Z . In this chapter, we focus on the classical setting where Z is discrete (so $\mathcal{Z}$ is countable and can be identified with the set of natural numbers without loss of generality) and the $X \rightarrow Z$ process is deterministic, i.e., $Z = f(X)$ for some function f ; we treat the case of a stochastic encoder for continuous compressed representations in Section 42.2.2. For simplicity (and without loss of optimality, as it turns out), we can represent $Z \rightarrow \hat{X}$ also by a deterministic function g which we call the decoding transform, so that $\hat{X} = g(Z)$.

As in lossless compression, we are interested in transmitting the source $X \sim P_X$ using the fewest number of bits. Commonly, this is done by entropy-coding Z under an entropy model P_Z^θ , so that the expected bit-rate is the cross-entropy (see Section 41.2):

$$\mathcal{R}(f, P_Z^\theta) = \mathbb{E}_{x \sim P_X, z=f(x)} [-\log P_Z^\theta(z)] = \mathbb{E}_{z \sim P_Z} [-\log P_Z^\theta(z)] = H[P_Z, P_Z^\theta]. \quad (41.13)$$

Exactly as in Eq. (41.11), the optimal choice of P_Z^θ is the marginal encoding distribution $P_Z(k) \stackrel{\text{def}}{=} \mathbb{P}(f(X) = k)$, for which the cross-entropy equals the entropy $H[Z]$. Additionally, we want to ensure that the lossy reconstruction preserves useful information in the original data, according to some criteria for “usefulness”. We make the standard choice of requiring a small average *distortion*,

$$\mathcal{D}(f, g) = \mathbb{E}[\rho(X, \hat{X})] = \mathbb{E}[\rho(X, g(f(X)))], \quad (41.14)$$

where $\rho : \mathcal{X} \times \hat{\mathcal{X}} \rightarrow [0, \infty)$ is a *distortion metric* which measures the deviation between a source and reconstruction pair, e.g., Hamming distance in the discrete case or squared error in the continuous case. Combining the above, we can evaluate the effectiveness of a lossy compression algorithm based on its *rate-distortion* (or “R-D”) performance given by the tuple $(\mathcal{D}, \mathcal{R})$, and our goal in lossy compression is to simultaneously minimize both costs with respect to our choice of the encoding, decoding functions (f, g) and the entropy model P_Z^θ . For a given source P_X and distortion metric ρ , what is the best possible rate-distortion performance one can expect? This is a key question in the theory of lossy compression known as *rate-distortion theory* [124, 1500], and the answer is given by a fundamental quantity $R(D)$ known as the *rate-distortion function*, or R-D function for short. For a given source X , distortion metric ρ , and threshold $D \geq 0$ of acceptable distortion, $R(D)$ equals the smallest possible bit-rate (bits per source sample) required to compress the source with an average distortion not exceeding D , regardless of the computation complexity of the compression procedure. The R-D function is defined by [1500]

$$R(D) \stackrel{\text{def}}{=} \inf_{Q_{\hat{X}|X} : \mathbb{E}_{P_X \times Q_{\hat{X}|X}} [\rho(X, \hat{X})] \leq D} I[X; \hat{X}], \quad (41.15)$$

where $Q_{\hat{X}|X}$ is any Markov kernel from $\mathcal{X}$ to $\hat{\mathcal{X}}$ (so that $Q_{\hat{X}|X=x}$ is a probability measure on $\hat{\mathcal{X}}$ for every $x \in \mathcal{X}$), and $I[X; \hat{X}]$ denotes the mutual information under the joint distribution $P_X \times Q_{\hat{X}|X}$, defined as the KL divergence between the joint and the products of its two marginals. We can conceptualize each $Q_{\hat{X}|X}$ as a black-box lossy compressor that maps any input x stochastically to an output $\hat{X} \sim Q_{\hat{X}|X=x}$, and $R(D)$ is obtained by minimizing the mutual information over all $Q_{\hat{X}|X}$ whose expected distortion is within the threshold D . In the same way entropy (Eq. (41.2)) characterizes the fundamental performance limit of lossless compression, the R-D function characterizes the fundamental performance limit of *lossy* compression.

Compared to lossless compression with entropy coding (Section 41.2), achieving the optimal R-D performance at a reasonable complexity has proved more difficult in practice. The remaining section will discuss various lossy compression algorithms with increasing sophistication. Instead of fixing a distortion threshold D and performing constrained optimization as in Eq. (41.15), it is often more convenient to minimize a scalar combination of both costs called the R-D Lagrangian [46, 167, 298]⁴

$$\mathcal{L}(f, g, P_Z^\theta; \lambda) = \mathcal{R}(f, P_Z^\theta) + \lambda \mathcal{D}(f, g), \quad (41.16)$$

where $\lambda \geq 0$ is a hyperparameter parametrising the R-D trade-off. Assuming perfect optimization, minimizing the above for a fixed λ results in a lossy compressor (f, g, P_Z^θ) whose R-D performance $(\mathcal{D}, \mathcal{R})$ is Pareto-optimal, and repeating this procedure for different λ values traces out a Pareto frontier commonly known as an *R-D curve*.

41.3.1 From Vector Quantization to Transform Coding

We now illustrate the general setup by deriving a basic lossy compression method, vector quantization (VQ). Let the data and reconstructions be d -dimensional real vectors ($\mathcal{X} = \hat{\mathcal{X}} = \mathbb{R}^d$) and the representation Z be some integer in $\mathcal{Z} = \{1, 2, \dots, K\}$ for some $K \in \mathbb{N}^+$. Suppose we transmit Z under a uniform entropy model; this can be done by transmitting $Z - 1$ in binary using at most $\lceil \log_2 K \rceil$ bits; e.g., we use the codewords $\{0_2 = 00, 1_2 = 01, 2_2 = 10\}$ if $K = 3$. The rate loss (Eq. (41.13)) is thus constant, and we seek encoding and decoding maps (f, g) that minimize the distortion (Eq. (41.14)) in terms of squared error. The solution is given by the coupled equations below

$$f(x) = \arg \min_{z \in \{1, 2, \dots, K\}} \|x - g(z)\|^2, \quad \forall x \in \mathcal{X}, \quad (41.17)$$

$$g(z) = \mathbb{E}[X | f(X) = z], \quad z = 1, 2, \dots, K. \quad (41.18)$$

We have essentially recovered the fixed-point iterations of the *Lloyd-Max* or *k-means* algorithm [1071]. The decoder g essentially serves as a lookup table whose outputs $\mu_k \stackrel{\text{def}}{=} g(k), k \in \mathcal{Z}$ have the common interpretation of *codebook vectors* or *cluster centroids*, and the resulting VQ algorithm maps any given x to its nearest cluster centroid $\hat{x} = g(f(x)) = \mu_{f(x)}$. Instead of fixing the entropy model, we can also incorporate its choice into the design of VQ, resulting in *entropy-constrained VQ* [298]. By optimizing the resulting R-D Lagrangian (Eq. (41.16)), the optimal compressor satisfies similar conditions to the above where the encoding map additionally takes into account the code length associated with each cluster index, the decoding map remains the same, and the optimal entropy model is given by the marginal distribution of the encoding $Z = f(X)$ (same as in Eq. (41.11)).

As with stream codes in lossless compression, by compressing data in increasingly long messages or blocks⁵ VQ can be shown to asymptotically achieve the optimal R-D performance given by the R-D function (Eq. (41.15)) [337]. However, this approach runs into the curse of dimensionality: as the block length (or data dimensionality) increases, an exponentially large number of codebook vectors is required to represent the source distribution well, making it exponentially more expensive to store the codebook vectors and to perform the nearest-neighbour search (Eq. (41.17)) to encode a single data point. As a result, VQ is typically only applied to compression problems where a relatively small K (and therefore low bit-rate) suffices, such as speech coding [1465].

Instead of vector quantization, we may consider *scalar quantization* where each dimension of the data vector is separately quantized and then entropy-coded, ignoring the statistical dependencies between the different dimensions. This is computationally much cheaper, albeit at the cost of much worse R-D performance and poor reconstructions which violate the statistical properties of the data. As a middle ground, we can first transform the data into a suitable representation with ideally independent dimensions, and then apply scalar quantization much more effectively. This is the approach taken by *transform coding* [655], and is the basis of virtually all

⁴Strictly speaking, Eq. (41.16) corresponds to the Lagrangian relaxation of a constrained optimization problem where we minimize the bit-rate $\mathcal{R}$ subject to a hard constraint on the distortion $\mathcal{D}$ (as in the definition of the R-D function, Eq. (41.15)), up to a constant that has no effect on the optimization.

⁵We extend the distortion metric to operate on length n blocks by considering the per-block average distortion: $\rho(x^n, \hat{x}^n) \stackrel{\text{def}}{=} \frac{1}{n} \sum_i \rho(x_i, \hat{x}_i)$.

popular compression algorithms for digital media today, such as images, audio, and video. In transform coding, a data source X is first transformed into a vector of *transform coefficients* $Y = \psi(X)$ via an *analysis transform* ψ ; then the coefficients are quantized, typically by coordinate-wise rounding, to obtain a discrete representation $Z = \lfloor Y \rfloor$, so the overall encoding map is $f = \lfloor \cdot \rfloor \circ \psi$. Traditionally, the analysis transform is chosen to be linear and invertible, so that the decoding map, also called the *synthesis transform*, is simply the inverse $g = \psi^{-1}$. A classic example is the Karhunen-Loève Transform (KLT), equivalently computed by Principal Component Analysis, which maps a given data point into the eigen-basis of the covariance matrix of X , resulting in decorrelated transform coefficients. Another example is the discrete cosine transform (DCT), which is shown to empirically approximate KLT asymptotically [14] and is one of the most widely used transforms in signal processing and compression (e.g., in the JPEG algorithm). The choice of a linear (and often orthogonal) transform simplifies the theoretical analysis and design of transform coding algorithms, with the KLT shown to be optimal for Gaussian sources [656].

Compared to VQ whose only effective constraint is the cardinality of the set of discrete representations, transform coding introduces additional structure into the design of the lossy compressor. As a result, transform coding generally cannot achieve the information-theoretically optimal R-D performance, but the lack of theoretical optimality is more than compensated for by its vastly improved computation efficiency over VQ, as evidenced by its wide-spread application.

41.3.2 Neural Lossy Compression

Having introduced the general problem of lossy compression and two prototypical algorithms, we now consider *neural lossy compression*, i.e., lossy compression using deep learning tools such as neural networks, typically learned end-to-end on the source data samples. Compared to earlier algorithms such as those based on linear transform coding, neural lossy compression can be optimized on custom performance metrics and better adapt to the source distribution, thereby offering more flexibility and improved compression performance [88]. Although neural networks have been used for lossy image compression since before the 1990s [844, 1537], techniques only emerged in the last decade that allowed them to be effectively trained end-to-end on a rate-distortion loss [83, 1596, 1618]. A major stumbling block has to do with *quantization*, by which we mean any function that maps from a continuous domain to a discrete one with countably (and often finitely) many values; an example is the VQ encoding map in Eq. (41.17). As neural networks and other smooth function approximators produce real-valued outputs, quantization is required to obtain discrete representations that can be entropy-coded. However, quantization does not have a useful derivative (zero almost everywhere), and the rate loss (Eq. (41.13)) is defined only with respect to discrete representations, which poses a significant challenge to gradient-based optimization. Different neural lossy compression methods over the years have tackled the challenge of quantization in different ways, including stochastic binarization [1618], straight-through estimator [1032], and continuous relaxations to VQ [12, 1166, 1805]. One approach based on uniform quantization [83, 1596] has proved particularly effective and become well-established. We will focus on this approach, known as *nonlinear transform coding* (NTC) since it generalizes transform coding (Section 41.3.1) to nonlinear transforms. We refer the reader to [1808, Section 3.3] and [88] for a comprehensive review.

In nonlinear transform coding, the lossy compressor has the same structure as in transform coding, where the discrete representation $Z = f(X) = \lfloor \psi(X) \rfloor$ is obtained by rounding the output of an analysis transform ψ , and the reconstruction $\hat{X} = g(Z)$ is decoded using a synthesis transform g . However, instead of linear ψ and g , we now consider richer, nonlinear transforms parametrised by deep neural networks, and also adopt a neural density estimator for the entropy model P_Z^θ similarly to in neural lossless compression (Section 41.2.5). Note that the assumption of a fixed quantization grid with bin widths equal to one is without loss of generality, assuming a sufficiently powerful pair of nonlinear transforms [84]. The neural network architectures for the nonlinear transforms vary by data modality and may incorporate domain knowledge in their design choices. For example, in image compression the transforms are commonly convolution neural networks [84, 85, 288, 1804] or vision transformers [1881], with spatial downsampling in the analysis and upsampling in the synthesis transforms. As another example, in low-latency video compression [13, 1084, 1802] the transforms receive the reconstruction of the previous video frame in addition to the current frame as input, and therefore can be seen as recurrent neural networks.

To address the challenge of end-to-end optimization posed by quantization, we replace rounding with additive uniform noise and consider a suitable choice of entropy model [83]. In particular, instead of the quantized representation $Z = \lfloor Y \rfloor \in \mathbb{Z}^m$ where $Y = \psi(X) \in \mathbb{R}^m$ is the vector of transform coefficients, consider the continuous “noisy” representation perturbed by uniform noise,

$$\tilde{Z} = Y + U, \quad U \sim \mathcal{U}([-0.5, 0.5]^m). \quad (41.19)$$

Here U is an independent random vector with uniform density on the unit hypercube $[-0.5, 0.5]^m$, which has the same width (one) as the quantization bin. Under certain conditions, independent uniform noise approximates the statistical properties of quantization error $Y - \lfloor Y \rfloor$, and additive uniform noise has been commonly used as a model for quantization in signal processing [118, 661].

Replacing Z by $\tilde{Z}$, the distortion loss (Eq. (41.14)) now becomes

$$\tilde{\mathcal{D}}(\psi, g) = \mathbb{E}[\rho(X, g(\tilde{Z}))] \stackrel{\text{def}}{=} \mathbb{E}_{x \sim P_X, y = \psi(x), u \sim \mathcal{U}([-0.5, 0.5]^m)}[\rho(x, g(y + u))], \quad (41.20)$$

which is now differentiable with respect to parameters of the encoding (and decoding) transform. However, the rate loss (Eq. (41.13)) is still only defined in terms of Z . It turns out that the additive uniform noise approximation also suggests a useful differentiable proxy for the rate loss. Denote the densities of $Y, \tilde{Z}$ and U by $p_Y, p_{\tilde{Z}}$ and p_U respectively, then the distributions of Z and $\tilde{Z}$ can be written in terms of p_Y as

$$P_Z(z) = \int_{[-0.5, 0.5]^m} p_Y(z + y) dy, \quad \forall z \in \mathbb{Z}^m, \quad (41.21)$$

$$p_{\tilde{Z}} = p_Y * p_U, \quad (41.22)$$

where $*$ denotes convolution and the density p_U is simply the indicator function for the unit hypercube $[-0.5, 0.5]^m$. Crucially, as illustrated in Fig. 41.2, the density of $\tilde{Z}$ interpolates the probability masses of Z on all integer grid points, and thus can be seen as a continuous relaxation of the latter. This observation suggests the following strategy for entropy modelling: instead of parametrising and learning P_Z (a high-dimension discrete distribution) directly, we learn a continuous density model $p_{\tilde{Z}}^\theta$ of the same form as $p_{\tilde{Z}}$; we can then easily obtain an entropy model P_Z^θ by evaluating $p_{\tilde{Z}}^\theta$ on integer grid points. Thus, we choose

$$P_Z^\theta(z) = \int_{[-0.5, 0.5]^m} p_\theta(z + y) dy, \quad \forall z \in \mathbb{Z}^m, \quad (41.23)$$

$$p_{\tilde{Z}}^\theta = p_\theta * p_U, \quad (41.24)$$

where p_θ is our choice of a neural density model over $\mathbb{R}^m$; we will discuss its choice in more detail below. Replacing Z by $\tilde{Z}$ and P_Z^θ by $p_{\tilde{Z}}^\theta$ in the rate loss (Eq. (41.13)), we obtain the following proxy rate loss which reduces to the cross-entropy between $p_{\tilde{Z}}$ and $p_{\tilde{Z}}^\theta$:

$$\tilde{\mathcal{R}} \stackrel{\text{def}}{=} \mathbb{E}_{X \sim P_X, u \sim p_U, \tilde{z} = \psi(x) + u}[-\log p_{\tilde{Z}}^\theta(\tilde{z})] = \mathbb{E}_{\tilde{z} \sim p_{\tilde{Z}}}[-\log p_{\tilde{Z}}^\theta(\tilde{z})]. \quad (41.25)$$

Combining Eqs. (41.20) and (41.25), the overall proxy R-D Lagrangian loss for nonlinear transform coding is then given by

$$\mathcal{L}_{NTC} \stackrel{\text{def}}{=} \mathbb{E}_{X \sim P_X, u \sim p_U, \tilde{z} = \psi(x) + u}[-\log p_{\tilde{Z}}^\theta(\tilde{z}) + \lambda \rho(x, g(\tilde{z}))], \quad (41.26)$$

which is differentiable with respect to the parameters of all model components (ψ, g, p_θ) and can be optimized end-to-end on data samples by stochastic gradient descent. After training, new data can then be compressed and decompressed following the same “analysis $\rightarrow$ round $\rightarrow$ entropy-code $\rightarrow$ synthesis” procedure as in transform coding. This results in a gap between additive noise during training and uniform quantization at test time. One way to reduce this gap is to directly optimize the choice of the discrete representation for each given data instance [1805], which is rooted in iterative amortized inference [1133] and variational optimization [1553]. Alternatively, at test time, one may replace uniform quantization with *dithered quantization* [11], i.e., quantization with a random offset; this essentially performs lossy compression with a stochastic code and will be discussed in Section 42.2.

We now elaborate on the choice of the entropy model further. Note that evaluating P_Z^θ at test time for entropy coding (and similarly $p_{\tilde{Z}}^\theta$ during training) still requires solving an integral (Eqs. (41.23), (41.24)) and can be computationally expensive. Therefore, in practice p_θ is chosen to be fully factorized so that the m -dimensional integral reduces to a product of univariate integrals. We can further avoid computing the univariate integrals by parametrising and learning a cumulative distribution function (CDF) instead of the density p_θ [85]: if $F_\theta : \mathbb{R} \rightarrow [0, 1]$ is differentiable, monotonically increasing, and satisfies $\lim_{\xi \rightarrow -\infty} F_\theta(\xi) = 0$ and $\lim_{\xi \rightarrow \infty} F_\theta(\xi) = 1$, then its derivative $\frac{\partial F_\theta(\xi)}{\partial \xi}$ is a valid probability density function. Parametrising the distribution of the i th dimension by a CDF model $F_\theta^i(\xi)$, then $P_{Z_i}^\theta$ (and similarly $p_{\tilde{Z}_i}^\theta$) can be conveniently computed by subtraction

$$P_{Z_i}^\theta(\xi) = F_\theta^i(\xi + 0.5) - F_\theta^i(\xi - 0.5). \quad (41.27)$$

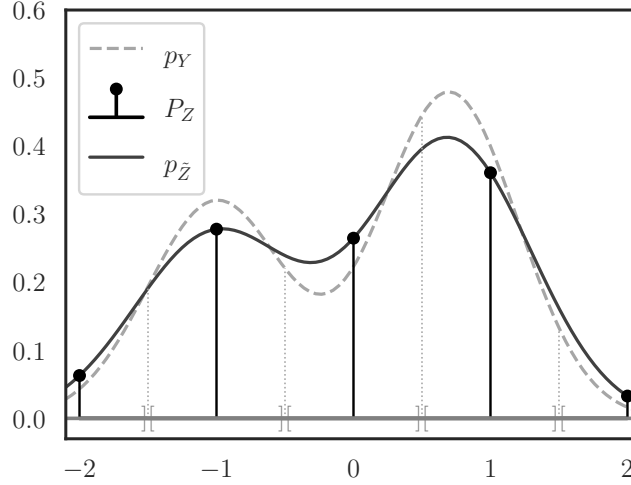


Figure 41.2: Relationship between the distributions of transform coefficients p_Y (dashed curve), quantized coefficients P_Z (vertical “lollipop” stems), and noise-perturbed coefficients $p_{\tilde{Z}}$ (solid curve), in one dimension. P_Z consists of a train of Dirac delta peaks, where the height of each peak is the probability of Y being quantized to the corresponding integer. $p_{\tilde{Z}}$ is obtained by convolving p_Y with uniform noise, and smoothly interpolates P_Z between all integer values.

Some common choices for F_θ are weighted combinations of logistic sigmoid functions [1452] and MLPs with suitable constraints [85]. To increase the expressiveness of the entropy model, state-of-the-art nonlinear transform coding approaches go beyond a fully factorized model and incorporate hierarchical modelling [85, 1183, 1809] and/or autoregressive modelling [1183, 1351].

41.3.3 Connection with latent variable models

There is a natural connection between lossy compression and latent variable modelling. Analogously to in lossless compression Section 41.2.5, we can give a lossy compression interpretation to a family of latent variable models trained by ELBO maximization. In some cases the correspondence is exact and the compression performance implied by the ELBO can be achieved with deterministic encoding, whereas more generally such models optimize an upper bound on the rate-distortion function and can be turned into lossy compressors with the more advanced techniques in Chapter 42.

Gaussian mixture model

As a first example, consider a Gaussian mixture model given by a prior model P_Z^θ and conditional likelihood model with density $p_\theta(x|z)$. Suppose the latent space $\mathcal{Z}$ consists of integers $\{1, 2, \dots, K\}$ and P_Z^θ is a categorical distribution parametrised by a probability vector $w \in \Delta^{K-1}$, and the likelihood model is given by a conditional density

$$p_\theta(x|k) = \mathcal{N}(x|\mu_k, \sigma^2 I), \quad k = 1, 2, \dots, K, \quad (41.28)$$

where $\{\mu_k \in \mathbb{R}^d : k = 1, 2, \dots, K\}$ are the means of the component Gaussians and $\sigma^2 I$ is a shared covariance matrix. We consider σ^2 fixed, so that the overall parameter tensor consists of $\theta = [w; \mu_1, \dots, \mu_K]$. The mixture model defines a marginal density over $\mathbb{R}^d$ via

$$p_\theta(x) = \sum_z P_Z^\theta(z) p_\theta(x|z) = \sum_{k=1}^K w_k \mathcal{N}(x|\mu_k, \sigma^2 I). \quad (41.29)$$

To estimate the model, it is common to minimize the negative ELBO (“NELBO”):

$$\mathbb{E}_{x \sim P_X} [-\log p_\theta(x)] \leq \mathbb{E}_{x \sim P_X, z \sim Q_{Z|X=x}} [-\log p_\theta(x|z)] + \mathbb{E}_{x \sim P_X} [\text{D}_{\text{KL}}(Q_{Z|X=x} \| P_Z^\theta)], \quad (41.30)$$

where $Q_{Z|X=x}$ is a variational distribution over $\mathcal{Z}$ for each x . We will see that the NELBO has an interpretation as the rate-distortion cost of lossy compression (Eq. (41.16)), as follows.

Plugging in the Gaussian density for $p_\theta(x|z)$, the first term of the NELBO simplifies to

$$\mathbb{E}_{x \sim P_X, z \sim Q_{Z|X=x}} [-\log p_\theta(x|z)] = \frac{1}{2\sigma^2} \mathbb{E}_{x \sim P_X, z \sim Q_{Z|X=x}} [\|x - \mu_z\|^2] + \text{const.} \quad (41.31)$$

This very much resembles a reconstruction error if we interpret the variational posterior as a stochastic encoder $x \rightarrow Q_{Z|X=x}$ and define the decoder g to act as a lookup table $g_\theta(k) = \mu_k, k \in \mathcal{Z}$. Note that we have first seen the negative ELBO in our analysis of lossless compression with latent variable models in Section 41.2.5. A major difference is in the interpretation of the likelihood term: in Eq. (41.12), $P_{X|Z=z}^\theta(x)$ is the probability of discrete data, and $-\log P_{X|Z=z}^\theta(x)$ is the lossless compression cost of the data under a conditional entropy model given z ; by contrast, here $p_\theta(x|z)$ is a probability density, and $-\log p_\theta(x|z)$ corresponds to the residual error between x and the lossy reconstruction $g_\theta(z)$.

The expected KL divergence term of the negative ELBO also has an interpretation as the “bit-rate” of lossy compression using the technique in Section 42.2. For now, note that if we restrict our variational posterior distributions to be deterministic, i.e., $Q_{Z|X=x} = \delta_{f(x)}$ for some function $f: \mathcal{X} \rightarrow \mathcal{Z}$, then the “rate” term of the NELBO reduces to the same cross-entropy term as in Eq. (41.12), i.e.,

$$\mathbb{E}_{x \sim P_X} [\text{D}_{\text{KL}}(\delta_{f(x)} \| P_Z^\theta)] = \mathbb{E}_{x \sim P_X} [-\log P_Z^\theta(f(x))] = \mathbb{E}_{x \sim P_X, z=f(x)} [-\log P_Z^\theta(z)], \quad (41.32)$$

thus its interpretation as the bit-rate is exact. Additionally, if we ensure a positive w and take the limit $\sigma^2 \rightarrow 0$, the NELBO reduces to purely a distortion loss, and the expectation-maximization (EM) algorithm [407]⁶ for the Gaussian mixture model reduces to the k -means algorithm [160]. In particular, the E-step of the EM algorithm computes the optimal variational posterior via Bayes’ rule:

$$Q_{Z|X=x}^*(k) \propto P_Z^\theta(k) p_\theta(x|k) = w_k \exp\left(-\frac{\|x - \mu_k\|^2}{2\sigma^2}\right), \quad k = 1, 2, \dots, K. \quad (41.33)$$

In the limit $\sigma \rightarrow 0$, a “winner-takes-all” phenomenon occurs where $Q_{Z|X=x}^*$ assigns probability 1 to the index of the closest cluster and reduces to the VQ encoder Eq. (41.17), and the M-step sets each μ_k to its respective cluster mean as in Eq. (41.18).

Variational Autoencoders

We now consider a broader family of latent variable models whose likelihood model is parametrised by an arbitrary decoding map g . When g and the approximate posterior family $Q_{Z|X}$ are parametrised by neural networks and jointly trained on the ELBO, such latent variable models are often known as *variational autoencoders* (VAEs) [892, 1391]. As in the Gaussian mixture example above, we denote the prior and likelihood models by P_Z^θ and $p_\theta(x|z)$, respectively. Now suppose that the conditional likelihood density is *aligned with* a distortion function ρ in the sense that

$$p_\theta(x|z) = C \exp\{-\lambda \rho(x, g_\theta(z))\}, \quad (41.34)$$

where $\lambda > 0$ and C are constants that do not depend on x or z . As a result, the negative log-likelihood density precisely corresponds to the distortion between x and its reconstruction $g_\theta(z)$, weighted by λ ,

$$-\log p_\theta(x|z) = \lambda \rho(x, g_\theta(z)) + \text{const}. \quad (41.35)$$

For example, a Gaussian $p_\theta(x|z) = \mathcal{N}(x|g_\theta(z), \frac{1}{2\lambda}I)$ is aligned with squared error distortion $\rho(x, \hat{x}) = \|x - \hat{x}\|^2$, and a Laplacian distribution is aligned with absolute error distortion. Thus, the NELBO (Eq. (41.30)) again has the interpretation of a rate-distortion loss, where the rate term reduces to cross-entropy when the encoder $Q_{Z|X}$ is deterministic. Similarly to lossless compression with bits-back coding (Section 42.1), (amortized) variational inference (i.e., optimizing the NELBO with respect to $Q_{Z|X}$) corresponds to finding the optimal stochastic encoder for a fixed choice of prior and conditional likelihood (equivalently, the decoder g_θ), and training the latent variable model by ELBO maximization corresponds to learning a lossy compressor of the data by minimizing a rate-distortion loss [84, 1805, 1807]. The correspondence is particularly salient in nonlinear transform coding (Section 41.3.2). We can reinterpret the uniform noise perturbation during training (Eq. (41.19)) as sampling from a particular form of stochastic encoder or variational posterior $Q_{\tilde{Z}|X}$, defined by

$$Q_{\tilde{Z}|X=x} = \mathcal{U}(\psi(x) - 0.5, \psi(x) + 0.5) \quad \forall x \in \mathcal{X}, \quad (41.36)$$

i.e., a uniform distribution over a hypercube centred on the vector of transform coefficients $\psi(x)$. The corresponding rate term works out to exactly equal the proxy rate loss in Eq. (41.25),

$$\mathbb{E}_{x \sim P_X} [\text{D}_{\text{KL}}(Q_{\tilde{Z}|X=x} \| p_{\tilde{Z}}^\theta)] = \mathbb{E}_{x \sim P_X} [-h[Q_{\tilde{Z}|X=x}]] + \mathbb{E}_{x \sim P_X, \tilde{z} \sim Q_{\tilde{Z}|X=x}} [-\log p_{\tilde{Z}}^\theta(\tilde{z})], \quad (41.37)$$

⁶See also Chapter 40 for an application of the EM algorithm.

where $h[\cdot]$ denotes differential entropy, and we used the fact that the uniform posterior in Eq. (41.36) has zero differential entropy. Thus, the R-D loss of nonlinear transform coding (Eq. (41.26)) exactly corresponds to the NELBO of a VAE specified by prior p_Z^θ as in Eq. (41.24), a distortion-aligned likelihood density $p_\theta(x|z)$ as in Eq. (41.34), and a variational posterior family $Q_{Z|X}$ as in Eq. (41.36). We note that unlike the fixed prior in the original VAE [892, 1391] and related work on Bayesian deep learning [174], the prior model in neural compression serves as an entropy model for the latents and is therefore learned jointly with other components of the lossy compressor for the best compression performance.

Empirically, the NELBO proxy of nonlinear transform coding (Eq. (41.26)) closely approximates the actual R-D performance achieved with quantization at test time [84]. This is ensured by design through specific restrictions on the prior and posterior distributions (Eq. (41.24), Eq. (41.36)) in the corresponding VAE. By comparison, in Bayesian deep learning it is common to adopt flexible variational posterior [1390] and prior distributions [1621] to improve the ELBO and the quality of the learned latent variable model. Naturally, we may wonder if an improved latent variable model also results in improved lossy compression performance, as in the case of lossless compression with bits-back coding [1625] (also see Section 42.1). While this is possible to some extent using iterative inference [1805] and posterior-adaptive quantization [1806], a general answer turns out to be non-trivial. Theoretically, *any* latent variable model with a distortion-aligned likelihood (Eq. (41.34)) optimizes a variational upper bound on the R-D function of the data source [1803, 1807], implying that there exists *some* lossy compressor that achieves the R-D performance given by the rate and distortion terms of the NELBO. Empirically, deep hierarchical VAEs have been demonstrated to yield tight upper bounds on the R-D function [1803]. However, it remains an open question how to convert such a latent variable model into a practical lossy compressor in an efficient manner. Our goal would be achieved if there exists an alternative to entropy coding that works with *any* stochastic encoder $Q_{Z|X}$ and prior P_Z^θ and can transmit samples from $Q_{Z|X=x}$ using a bit-rate equal to the expected KL $\mathbb{E}_{x \sim P_X} [\text{D}_{\text{KL}}(Q_{Z|X=x} \| P_Z^\theta)]$, averaged over $x \sim P_X$. This is indeed possible, although typically with a logarithmic overhead to the KL divergence, through a technique known as *channel simulation* or *relative entropy coding*. Developing efficient algorithms in this setting is an active research topic that we will explore in Section 42.2.

Chapter 42

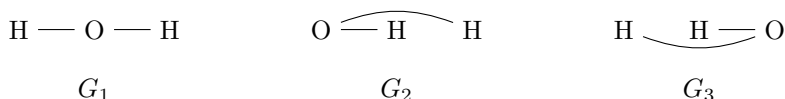
Learned Compression with Stochastic Codes

In the previous chapter, we described how classical source coding theory can be combined with modern machine learning techniques to achieve better data compression. However, we also saw that entropy codes such as Huffman coding (Section 41.2.3) and Arithmetic Coding (Section 41.2.4) operate on discrete data by design. Discreteness meant that it was non-trivial to combine them with neural models. For lossless neural compression using the two-part code presented in Section 41.2.5, it limited our choice of the probabilistic model by fixing the variational posterior over our latent variables to be deterministic (see Equation 41.12 and the discussion below it). For lossy compression, we were forced to quantise the data to ensure discreteness. However, since quantization is not differentiable, we had to introduce continuous approximations to quantisation to train lossy neural compressors in Section 41.3.2.

In this chapter, we describe two alternative approaches – bits-back coding for lossless and relative entropy coding for lossy compression – which lift all restrictions mentioned above. Therefore, these methods unlock a much broader class of probabilistic models for learned data compression. As we shall soon see, the high-level trick in both cases is to *randomise* the data encoding process in a controlled way. However, the benefits of these methods come at a price – implementing them can be more challenging than classical compression methods, and in some cases their runtime can be much worse than the comparable entropy coding algorithm; we will discuss these limitations at the end of their respective sections. Finally, we recommend the tutorial paper by [531] for a more detailed introduction and the excellent monograph of [1024] for a comprehensive treatment of relative entropy coding.

42.1 Lossless Compression with Randomization: Bits-back Coding

In Section 41.2, we saw how we can losslessly compress data $X \sim P_X$ using a probability model P_θ . However, in some more advanced cases, building an efficient probability model might be challenging, or even computationally intractable. Perhaps the simplest example is when we are interested in encoding unordered data or data up to some symmetry. Molecular structures are a prime example, since they often exhibit symmetries between the atomic bonds. For example, the atomic structure of water, H_2O , admits the labelled graph representation $\text{H} - \text{O} - \text{H}$. However, the following are all valid *ordered* representations of the same graph:



$$G_1 = ((\text{H}, \text{O}, \text{H}), \{\{1, 2\}, \{2, 3\}\}), \quad (42.1)$$

$$G_2 = ((\text{O}, \text{H}, \text{H}), \{\{1, 2\}, \{1, 3\}\}), \quad (42.2)$$

$$G_3 = ((\text{H}, \text{H}, \text{O}), \{\{1, 3\}, \{2, 3\}\}) \quad (42.3)$$

The above examples show that we need to take care that our compression algorithm built for labelled graphs does not encode more information than necessary: ideally, we would like to build a compressor that assigns the *same*

code to all equivalent representations of our data. In this section, we describe bits-back coding, which provides an elegant way of achieving this goal using randomisation. Moreover, in 42.1.4, we combine bits-back coding with latent variable models to build efficient lossless data compressors for any type of discrete data.

42.1.1 Invertible Sampling and Asymmetric Numeral Systems

The first fundamental concept bits-back coding relies on is the duality between data compression and random sampling,¹ which we first illustrate at an intuitive level. Consider a stream code, such as arithmetic coding (Section 41.2.4), which has the following components: for any probabilistic model P over the symbols to be encoded, and a pre-existing message/stream m , let $\text{enc}_P(x, m) = m'$ be the function that encodes the symbol x into m , producing the updated message/stream of bits m' . Here, we consider only entropy codes, meaning that the message length increases by roughly the self-information of x : we have $|m'| - |m| \approx -\log_2 P(x)$. Finally, a stream code also has a corresponding decoding function $\text{dec}_P(m') = (x, m)$, which is the inverse of the encoder:

$$\text{dec}_P(\text{enc}_P(x, m)) = (x, m) \quad (42.4)$$

Now, let $X \sim P$ be a discrete random variable and, starting with the empty message ϵ , consider encoding X and setting $M = \text{enc}_P(X, \epsilon)$. A crucial feature of entropy codes is that the bits of M ought to be uniformly distributed. Intuitively, this holds since if we know the true distribution of the source, then entropy codes extract all statistical structure from X and produce incompressible uniform noise. Specifically, if we let F be the CDF of P , we can view arithmetic coding directly as the probability integral transform, since it encodes X as a rational approximation of $F(X)$ [337, Chapter 13.3].

Given the above, consider the reverse scenario: let U be a fair Bernoulli process, i.e. an infinite bitstring consisting of iid fair coin flips. Moreover, assume for the moment that dec_P is a total function, that is, its output is well-defined for any binary string (with some effort, arithmetic coding can be implemented for this to hold). If we “decode” U and set $(X', U') = \text{dec}_P(U)$, what is the distribution of X' ? Well, since dec_P is the inverse of enc_P , and we saw above that for $X \sim P$ we have that $\text{enc}_P(X)$ is uniform, we therefore must have that $X' \sim P$. This beautiful identity has the following intuitive interpretation: we can view U as the “seed” of the pseudo-random number generator (PRNG) dec_P , with enc_P as its inverse, which, given X' (and the updated PRNG seed U') returns the random seed U that generated X' . Therefore, we may view stream codes as *invertible samplers*. Here we glossed over several technical details, we refer the interested reader to [906] for a rigorous development of the above ideas.

A key assumption of the above argument was that the decoder of the stream code be a total function over binary strings. While arithmetic coding can be set up so that it fulfils this requirement, this is a non-trivial undertaking in practice. Furthermore, as we will see soon, it has another feature that makes it an unsuitable choice for bits-back coding: it encodes messages in a first-in-first-out (FIFO) structure, that is, if we encode a message $x_1 x_2 x_3 \dots$ into a stream m , the decoder recovers the symbols in the same order. A solution to both of these issues is to use a different stream code, *asymmetric numeral systems* [ANS; 459], instead. We refer the interested reader to Robert Bamler’s excellent exposition of ANS [89] – for our exposition of bits-back coding here, it suffices to know that the ANS decoder is trivially seen to be a total function on bitstrings and ANS encodes messages in a last-in-first-out (LIFO) structure, i.e., if we encode the message $x_1 x_2 \dots x_n$, the decoder recovers them in the order $x_n \dots x_2 x_1$.

42.1.2 A very brief introduction to symmetries

Before we can define compression schemes that eliminate the coding inefficiencies that arise from symmetries, we need to define what we mean by symmetry. Intuitively, a symmetry of an object is a transformation whose outcome does not change the object, such as rotating a square by 90 degrees. In this section, we develop the bare minimum mathematical machinery to capture the most important aspects of this intuition.

To this end, consider the set S_n , the so-called *symmetric group of degree n* , consisting of all invertible functions on n items. The elements of S_n are called *permutations*, and can be thought of as the possible orderings of n items. The power of permutations is that it turns out that any discrete symmetry can be represented using them.² Hence, for the mathematics to follow, we will assume henceforth that all abstract transformations are “implemented” using permutations. Using this insight, we make the following abstract definitions of the symmetries of objects, and leave it up to the practitioner to define the precise “implementation” of the symmetry.

¹Indeed, this duality is so important that it is arguably the central concept underpinning all of algorithmic information theory [1031], but we will not discuss this any further here.

²This is known as Cayley’s theorem.

Definition 42.1 (Permutable class). For $n \in \mathbb{N}$, a permutable class of order n is a set $\mathcal{F}$, together with a binary operation $\cdot : S_n \times \mathcal{F} \rightarrow \mathcal{F}$, called a group action.³

Intuitively, a permutable class defines the set of *ordered* objects/structures under consideration. Recalling the molecular structure example from earlier in the section, we may pick our permutable class $\mathcal{F}$ (of order three) as the set of labelled, ordered graphs with at most three vertices. Then, for an ordered representation of a graph, we may “implement” the group action as permuting the ordering of the nodes with the edges and labels changed appropriately. Concretely, one way to represent ordered, labelled graphs on a computer is using a labelled adjacency matrix:

$$G_1 \rightarrow A = \begin{matrix} & \begin{matrix} H & O & H \end{matrix} \\ \begin{matrix} H \\ O \\ H \end{matrix} & \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{matrix} \quad (42.5)$$

where if $A_{ij} = 1$ indicates that node i is connected to node j . Then, we could define the action of a permutation f on a graph as permuting the rows and columns (alongside the labels) of its adjacency matrix: $(f \cdot A)_{i,j} = A_{f^{-1}(i), f^{-1}(j)}$. For example, applying the function f defined as $f(1) = 2, f(2) = 3$ and $f(3) = 1$ to the adjacency matrix of water above, this would result in:

$$f \cdot G_1 = f \cdot \begin{matrix} & \begin{matrix} H & O & H \end{matrix} \\ \begin{matrix} H \\ O \\ H \end{matrix} & \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{matrix} = \begin{matrix} & \begin{matrix} H & H & O \end{matrix} \\ \begin{matrix} H \\ O \\ H \end{matrix} & \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{matrix} = G_3 \quad (42.6)$$

However, these adjacency matrices represent the same graph, as we can verify from the corresponding representations G_1 and G_3 above. But what systematic computational procedure could we use to check automatically whether two elements $x, y \in \mathcal{F}$ are the same? Since we are assuming that our objects are ordered in some way, the idea is to *sort* x and y . Then, denoting the sorted versions as $\bar{x}$ and $\bar{y}$, respectively, we could say that x and y are equivalent if $\bar{x} = \bar{y}$. We formalise this idea below.

Definition 42.2 (Canonical ordering, equivalence). Given a permutable class $\mathcal{F}$ of order n , a canonical ordering is any function $\mathbf{sort} : \mathcal{F} \rightarrow \mathcal{F} \times S_n$, such that:

- For all $x \in \mathcal{F}$, we get a sorted representation and a sorting permutation $(\bar{x}, s) = \mathbf{sort}(x)$ such that $s \cdot \bar{x} = x$. As a shorthand, we will use $\bar{\cdot}$ to denote the sorted element of $\mathcal{F}$ that $\mathbf{sort}(\cdot)$ returns.
- For $x, y \in \mathcal{F}$, we say that x and y are equivalent if $\bar{x} = \bar{y}$.

Now, we use canonical orderings to define two important sets related to any $x \in \mathcal{F}$: the *equivalence class* $[x]$ and the *coset* S_n^x .

Definition 42.3 (Equivalence class, coset). For an element x of a given permutable class $\mathcal{F}$, the equivalence class of x is defined as the set of all elements of $\mathcal{F}$ equivalent to x :

$$[x] = \{y \in \mathcal{F} \mid \bar{x} = \bar{y}\} \quad (42.7)$$

Then, the coset⁴ S_n^x of x is the set of corresponding permutations $\mathbf{sort}$ returns for each element of $[x]$:

$$S_n^x = \{s \in S_n \mid (s, \bar{y}) = \mathbf{sort}(y), y \in [x]\} \quad (42.8)$$

Using the water example once more, we have that

$$[G_1] = \{G_1, G_2, G_3\} \quad (42.9)$$

We may view the equivalence classes’ bracket notation $[x]$ as a map that takes an ordered structure x and “forgets” its ordering. The final object we require to define bits-back coding is the set (more precisely, the group) of all transformations that leave a particular element of a permutable class invariant. As such, it is the set of all symmetries that an object possesses.

³Technically, a *left* group action.

⁴Technically, the *left* coset. While this definition is sufficient for our purposes here, we note that it is related to but does not match the usual group-theoretic use of the term.

Algorithm 33: Shuffle encoder

Input : stream m ,
 object x ,
 probability distribution P_θ

Output: updated stream m''

```
// Note that  $s \sim \text{Unif}(S_n^x)$ 
 $(s, m') \leftarrow \text{dec}_{\text{Unif}(S_n^x)}(m)$ 
 $y \leftarrow s \cdot \bar{x}$ 
return  $\text{enc}_{P_\theta}(y, m')$ 
```

Algorithm 34: Shuffle decoder

Input : stream m'' ,
 probability distribution P_θ

Output: decoded object y ,
 remaining stream m

```
 $(y, m') \leftarrow \text{dec}_{P_\theta}(y, m'')$ 
 $(\bar{y}, s) \leftarrow \text{sort}(y)$ 
 $m \leftarrow \text{enc}_{\text{Unif}(S_n^y)}(s, m')$ 
return  $y, m$ 
```

Definition 42.4 (Automorphism group). *For an element x of a permutable class $\mathcal{F}$ of order n , its automorphism group is the set of all permutations that do not change x :*

$$\text{Aut}(x) = \{s \in S_n \mid s \cdot x = x\} \quad (42.10)$$

For example, $\text{Aut}(\text{H} - \text{O} - \text{H}) = \{(x \mapsto x), (x \mapsto 4 - x)\}$ is the automorphism group of the water graph.

Let us briefly take stock of what we have so far: we have identified the notion of ordered structures with permutable classes. We have also identified the notion of unordered structures with equivalence classes of isomorphic elements of a permutable class. We defined canonical orderings, using which we can “sort” elements of a permutable class and the automorphism group of an element being the set of all permutations that leave an object unchanged.

42.1.3 Bits-back coding with discrete symmetries: Shuffle Coding

We are now ready to derive the simplest form of bits-back coding, called shuffle coding [950]. The two key ideas of shuffle coding are these: 1) given an ordered structure x (i.e., the element of a permutable class), we encode a *random, but equivalent* ordered representation y ; and 2) we can use ANS as an invertible sampler to use a pre-existing stream m for the randomisation. Indeed, this is an almost complete description of shuffle coding, and we will spend the rest of this section unpacking it.

First, let us explain the randomisation more carefully. Assume the encoder receives an element x of a permutable class, and they wish to discard all order structure present in x , which they can achieve using the following strategy. They sort x to obtain $\bar{x}$, and draw a uniformly random permutation from x ’s coset $s \sim \text{Unif}(S_n^x)$ and compute $y = s \cdot \bar{x}$. Then, y will be a uniformly random element of x ’s equivalence class $[x]$, and thus the encoder succeeded in discarding the order structure. Nonetheless, y still has some order, albeit random, so encoding it will be no cheaper than encoding x directly. However, note that the decoder has the following “error-correction” ability: after receiving any $y \in [x]$, they can obtain the same $\bar{x}$ the encoder had by computing $\text{sort}(y)$.

We combine this capability with the second idea of using ANS as an invertible sampler to get an efficient coding algorithm: the encoder can use the pre-existing message to randomise $\bar{x}$. Concretely, assume we have some stream of bits m into which we wish to encode x using the ANS encoder using P_θ as the coding distribution. However, before doing so, we use the ANS decoder to set $(s, m') = \text{dec}_{\text{Unif}(S_n^x)}(m)$. Recall from Section 42.1.1, that if m consists of uniformly random bits (and empirically, this is roughly true) then we will have $s \sim \text{Unif}(S_n^x)$, as we require above. Furthermore, since we decoded from the stream, the total message length decreases, that is, $|m'| \leq |m|$. Next, we use our decoded s to compute $y = s \cdot \bar{x}$ as above, and we encode y into the updated stream m' to obtain $m'' = \text{enc}_{P_\theta}(y, m')$. We see that this procedure already produces a shorter overall message length than encoding x into m directly. However, before analysing the gain, we need to demonstrate that this procedure is invertible, otherwise it would not be of much use.

Thus, consider the decoder, who receives m'' from above. Using P_θ , they use the ANS decoder to recover $(y, m') = \text{dec}_{P_\theta}(m'')$. But they can do more: they can recover the encoder’s permutation s by sorting y : $(\bar{y}, s) = \text{sort}(y)$. Then, since x and y are equivalent by construction, we also have $S_n^x = S_n^y$ and hence the decoder can re-encode s into m' to obtain the original message $m = \text{enc}_{\text{Unif}(S_n^y)}(s, m')$. This encoding-decoding procedure constitutes bits-back coding, and we summarise the procedure in Algorithms 33 and 34.

Now that we see that the bits-back coding procedure is correct, what is its coding cost? One important aspect we haven’t mentioned yet, is what coding distribution P_θ we ought to use. A crucial desideratum is that P_θ be exchangeable, i.e. for $x, y \in \mathcal{F}$ if x and y are equivalent, we should have $P_\theta(x) = P_\theta(y)$. Indeed, it can be shown that for any distribution $\tilde{P}_\theta$ over the equivalence classes, $\tilde{P}_\theta$ induces a unique exchangeable distribution P_θ over

$\mathcal{F}$ [950, Lemma 3.1]. Furthermore, it can be shown by applying the so-called orbit-stabiliser theorem from group theory, that $|S_n^x| = n!/|\text{Aut}(x)|$. Hence, $\text{Unif}(S_n^x)$ assigns $|\text{Aut}(x)|/n!$ probability to each $s \in S_n^x$. Putting these together with the fact that ANS is an entropy code, we find that decoding a uniformly random permutation from the coset of x reduces the message length by approximately $\log_2 n! - \log_2 |\text{Aut}(x)|$ bits, while encoding y using P_θ increases the message length by approximately $-\log_2 P_\theta(y)$ bits. Thus, in total the cumulative message length increase is

$$|m''| - |m| \approx -\log_2 P_\theta(y) - \log_2 n! + \log_2 |\text{Aut}(x)| \text{ bits}, \quad (42.11)$$

which we see is equal to directly encoding $[x]$ using $\widetilde{P}_\theta$.

Shuffle coding is the state-of-the-art method for optimally encoding objects with discrete symmetries. Indeed, Kunze et al. [949] achieve an impressive 46% reduction in the codelength when compressing molecules and a 26% reduction when compressing graphs of certain social networks. However, there are several implementation challenges that one must take into account in practice. In the rest of this section, we discuss the most salient ones and provide some references for further reading.

The biggest outstanding issue of shuffle coding as we presented here is that its efficiency relies on so-called “clean, initial bits.” Namely, we need to have a non-empty bitstream m to begin with, otherwise the encoder cannot decode the random permutation at the start of the encoding process. In other words, applying shuffle coding to a single object x will actually cost $-\log_2 P_\theta(y)$ bits, as we need to provide an initial stream $-\log_2 n! + \log_2 |\text{Aut}(x)|$ bits so that the shuffle encoder (Algorithm 33) can decode an element of x ’s coset. Thus, shuffle coding in this form is best applied for compressing entire datasets of unordered objects, because this way the initial overhead in bits amortises over the dataset. Fortunately, it also turns out that shuffle coding can be modified so that it can also encode single unordered objects; we refer the interested reader to Kunze et al. [949] for the details. Furthermore, we need that the bits of m be (roughly) uniformly distributed, otherwise the coding efficiency of bits-back coding can suffer. Fortunately, Townsend et al. [1625] show that the message streams produced by shuffle coding are indeed roughly uniform and thus the coding cost is close to optimal in practice. Note also that, as we saw above, in the optimal case $|\text{Aut}(x)| = 1$, since we make the largest coding gain this way. However, this also means that naively implementing shuffle coding in this case requires $O(\log_2 n!) = O(n \log_2 n)$ initial bits. Fortunately, Kunze et al. [949] show that this can be reduced to $O(\log_2 n)$ bits.

The other big issue with implementing shuffle coding is computing the canonical ordering `sort`. There is currently no known general method to construct a canonical ordering for a given permutable class $\mathcal{F}$. Indeed, it is not even known whether a canonical ordering exists for every permutable class. Thus, at present the implementation of `sort` must rely on the specific properties of the precise underlying class. Unfortunately, even if a canonical ordering is known for a certain class, it might be difficult to compute. For example, in the case of arbitrary unlabelled graphs, there is no known polynomial-time algorithm for computing their canonical ordering or automorphism group. We also omitted implementation details for designing P_θ and drawing samples from $\text{Unif}(S_n^x)$. A detailed discussion on all of these topics can be found in Kunze et al. [949, 950].

42.1.4 Bits-back coding with side-information

The previous section described how to construct coding algorithms for unordered data using shuffle coding, a special case of bits-back coding. This section describes another application of bits-back coding: building lossless compression algorithms using essentially arbitrary latent variable models. Therefore, bits-back coding enables us to build lossless neural data compression algorithms even from probabilistic models with continuous latent variables such as variational autoencoders and normalising flows [754, 1625], thereby removing most limitations we discussed in Section 41.2.5.

To begin, let us restate the main mechanism that made shuffle coding in Section 42.1.3 possible. The key observation is that if the encoder sorts the data and then randomly permutes it, the decoder can sort the data themselves to recover not just the sorted version of the data, but also the random permutation the encoder used. The way shuffle coding exploited this observation was to use ANS (Section 42.1.1) to sample/decode the random permutation from the pre-existing message stream to reduce its length and then proceeded to encode the randomised data into the reduced-length stream.

Abstractly, writing X for the sorted data and Y for the permuted data, we can restate the above by saying that for any $Y \sim P_{Y|X}$, the shuffle coding decoder can always recover X from Y . Then, the question is: can we generalise this procedure to different settings? For example, could we make the same idea work when Y is a real-valued vector? Excitingly, the answer is yes, and this section describes a very general approach to do this.

We start with generalising the scheme to arbitrary discrete spaces and distributions. The key idea is beautifully simple: since in general, the decoder cannot recover X from Y , instead of just encoding the randomised data Y ,

the sender encodes X, Y jointly. Thus, we construct the updated, generalised bits-back algorithm as follows. We start with a joint distribution $P_{X,Y} = P_X \times P_{Y|X}$ over the original data X and its randomised variant Y . Then, **the sender**, having pre-existing message stream m and upon receiving X :

1. Decodes $(Y, M') = \text{dec}_{P_{Y|X}}(m)$ using the conditional $P_{Y|X}$. Then, $Y \sim P_{Y|X}$ and the message length is reduced by approximately $-\log_2 P_{Y|X}(Y)$ bits.
2. Encodes X into M' using the conditional $P_{X|Y}$: it sets $M'' = \text{enc}_{P_{X|Y}}(X, M')$, increasing the message length by approximately $-\log_2 P_{X|Y}(X)$ bits.
3. Encodes Y into M'' using the marginal P_Y : it sets $M''' = \text{enc}_{P_Y}(Y, M'')$, increasing the message length by approximately $-\log_2 P_Y(Y)$ bits.

As before, note that this generalised procedure is also invertible, meaning that upon receiving M''' , the decoder can always recover the data X and the original stream M . Concretely, they:

1. Decode $(Y, M'') = \text{dec}_{P_Y}(M''')$ using the marginal P_Y .
2. Since Y is now known, decode $(X, M') = \text{dec}_{P_{X|Y}}(M'')$.
3. Since X is now known, re-encode Y into M' using the conditional $P_{Y|X}$ to recover the original stream: $m = \text{enc}_{P_{Y|X}}(Y, M')$.

Having convinced ourselves that the scheme is invertible, let us compute its efficiency. We saw that the encoding process first decreases the message length by approximately $-\log_2 P_{Y|X}(Y)$ bits, then increases it by $-\log_2 P_{X|Y}(X)$ and $-\log_2 P_Y(Y)$ bits, respectively. Thus, the total increase is approximately

$$\log_2 P_{Y|X}(Y) - \log_2 P_{X|Y}(X) - \log_2 P_Y(Y) = -\log_2 \frac{P_{X,Y}(X, Y)}{P_{Y|X}(Y)} = -\log_2 P_X(X) \quad (42.12)$$

which is the same efficiency as using entropy coding directly. This finally completes the picture from Section 41.2.5, showing that it is possible to achieve the theoretically optimal compression rate by fully eliminating the slack term in Equation 41.12.

So far, we have dealt solely with discrete spaces. However, as we explain next, bits-back coding is also applicable to “sufficiently nice” continuous distributions. Concretely, we will still assume that the noise-free version of the data X is discrete, but let Y be $\mathbb{R}^n$ -valued. To make bits-back coding work in this setting, we need to make the additional assumption that the support of Y ’s distribution can be quantised well. Concretely, let $p(y | x)$ and $p(y)$ be the probability densities of $P_{Y|X}$ and P_Y , respectively. Furthermore, for a positive real number δ let $\mathcal{Q}_\delta$ be a quantiser such that $\mathbb{P}(\mathcal{Q}_\delta(Y) = y | X = x) \approx p(y | x)\delta^n$ and $\mathbb{P}(\mathcal{Q}_\delta(Y) = y) \approx p(y)\delta^n$. Then, using the quantised density functions to decode/encode Y , Equation 42.12 tells us that applying the discrete bits-back procedure from above will increase the message length by

$$|M'''| - |M| \approx -\log_2 \frac{P_{X,Y}(X, Y)}{P_{Y|X}(Y | X)} \approx -\log_2 \frac{P_{X|Y}(X | Y)p(\mathcal{Q}_\delta(Y))\delta^n}{p(\mathcal{Q}_\delta(Y) | X)\delta^n} = -\log_2 P_X(X) \quad (42.13)$$

This means that we still retain efficiency at the entropy coding level. This might make it seem like the level of discretisation is arbitrary, so long as our quantisation assumption is satisfied. However, in practice, we usually want to set the discretisation level as low as possible, since it determines the amount of initial bits necessary to run the compressor. As an example, consider using 16-bit quantisation for compressing 1024×1024 greyscale images. To do this, we need on the order of $2^{20} \times \log_2 16 = 2^{22}$ bits, or roughly 524 kilobytes of initial bits.

Given the bits-back machinery we developed in this section, constructing lossless neural compressors is fairly straightforward using the two-part code construction we gave in Section 41.2.5. Indeed, after bits-back coding was invented in the early 1990s by Hinton and Van Camp [751], it quickly found use for compression with latent variable models, first implemented for a limited model class as early as 1997 by Frey and Hinton [560], who used Arithmetic Coding as the invertible sampler (ANS hadn’t yet been invented at this point). However, it was Townsend et al. [1625] who produced the first practical implementation of the bits-back coding technique using ANS for lossless neural compression with variational autoencoders. Since then, their method has been improved upon, scaled up and extended in several interesting directions; we refer the interested reader to the excellent papers of Ho et al. [754], Kingma et al. [896], Townsend et al. [1626].

42.2 Lossy Compression with Randomization: Relative entropy coding

After considering lossless compression in Section 42.1, we now treat lossy compression with stochastic codes in this section. Recall that Section 41.3 introduced quantisation as the fundamental mechanism for discarding information. While quantisation has the advantage that it is simple to implement, and has many decades of dedicated research and industrial practice behind it, recent advances in neural compression have shone light on some of its weaknesses too. First, quantisation is a non-differentiable operation, and hence, as we have already seen in Section 41.3.2, constructing efficient learned lossy compression models involves some non-trivial approximations. More importantly, quantisation error in general is difficult to control practically and analyse theoretically.

Thus, in this section we describe a different lossy compression mechanism: random perturbation. In its simplest form, the idea is to add some independent noise ϵ to our signal X to form the lossy signal $Y = X + \epsilon$. Note that ϵ , and hence Y may remain continuous, and the additive form for Y above immediately suggests to use the reparametrisation trick for gradient-based learning (see Section 13.2.1). Naturally, the question is how to compress Y . The answer is *relative entropy coding* (also called channel simulation, reverse channel coding and channel synthesis in various parts of the information theory literature [531, 1024]). In comparison to quantisation and entropy coding (Section 41.3), relative entropy coding algorithms are generally significantly more complex to implement. However, their benefit goes beyond avoiding the non-differentiability issues quantisation suffers from. Since we get to *pick* the distribution of Y in relative entropy coding (REC), we can develop more advanced features on top of it, such as differential privatisation or leveraging generative modelling to improve the perceptual quality of the compressor’s output; we will discuss these in Section 42.3.

42.2.1 The most special case: dithered quantisation

There is undoubtedly a large conceptual hurdle for any newcomer to REC: it is rather counter-intuitive that the stated goal of encoding a random sample from a possibly continuous space is even possible using only finitely many bits. Hence, the goal of this section is to give the reader a first idea of an algorithm.

First, for all our algorithms to work, we need a crucial new ingredient in the framework: shared randomness. From a theoretical perspective, shared randomness comprises some random variable (or collection of variables) W whose realisations are available to both the encoder and the decoder, akin to entangled quantum states.⁵ Practically, shared randomness may be achieved in various ways. Perhaps the simplest approach is for the encoder to use a pseudo-random number generator (PRNG) with a seed s that they encode in the preamble of their message (the first thing the receiver decodes), so that the decoder can use the same seed s for their PRNG and thus generate the same samples as the encoder did.

With this additional technical detail in mind, we now describe *dithered quantisation*, which takes some real-valued input signal $X \sim P_X$ and encodes a version of the input corrupted by independent uniform noise: $Y = X + \epsilon$, where $\epsilon \sim \text{Unif}(-1/2, 1/2)$. To achieve this, we make use of the following beautiful, yet little-known distributional identity: for any $c \in \mathbb{R}$ and two independent and identically distributed uniform random variables $U, U' \sim \text{Unif}(-1/2, 1/2)$, it holds that

$$\lfloor c + U \rfloor - U \sim c + U' \quad (42.14)$$

We refer to Equation 42.14 as the *dithering identity*, and it holds the key to our efficient coding scheme. Concretely, we set $W \leftarrow U \sim \text{Unif}(-1/2, 1/2)$ to be the shared randomness between the encoder and the decoder. Then, upon receiving the signal $X \sim P_X$, the encoder computes the perturbed and quantised signal $K = \lfloor X + U \rfloor$. Since K is discrete, we can use an entropy code to encode it (such as arithmetic coding, see Section 41.2.4); the optimal coding distribution turns out to be also the most natural choice: for $K = k$ we should use⁶

$$P(k \mid U) = \mathbb{P}(\lfloor X + U \rfloor = k \mid U) = P_X([k - U - 1/2, k - U + 1/2)). \quad (42.15)$$

Note, that conditioning on U here is fine, since we assume that it is shared and hence the receiver knows it. Thus, the receiver first recovers K using their knowledge of U and then computes $Y = K - U$. By Equation 42.14, we will have $Y \sim \text{Unif}(X - 1/2, X + 1/2)$.

The efficiency of dithered quantisation is also quite remarkable: using $P(k \mid U)$ as the coding distribution

⁵Indeed, REC was first conceived as part of investigations of quantum entanglement-assisted communication [117, 1561].

⁶For the second equality, we assumed the definition $\lfloor x \rfloor = \lfloor x + 1/2 \rfloor$; for other conventions the second equality changes accordingly.

above, we find that the bitrate is given by the mutual information between X and Y :⁷

$$H[Y | U] = H[K | U] = I[X; K - U] = I[X; Y] \quad (42.16)$$

We emphasise that the Shannon entropy $H[Y | U]$ above is not a mistake: even if Y itself is continuous, its description length *given the common randomness* is finite! Equation 42.16 indicates the significant gain we obtain using the shared randomness: without conditioning, $H[Y]$ would be infinite since Y is real-valued. In fact, in Section 42.2.2 we show that the coding rate $I[X; Y]$ is optimal, and as such it plays a similar role to the entropy in the classical, deterministic setting.

42.2.2 General Relative Entropy Coding and Its Limitations

In the previous section, we saw that we can leverage shared randomness to encode an additive uniform perturbation. In this section, we zoom out as much as possible and define REC in its greatest generality, and discuss its limitations.

As we already noted, the core idea of REC is to perturb the source X . Thus, if we denote by Y the output of the procedure, we may choose an arbitrary conditional distribution $P_{Y|X}$. Then, the idea is that upon receiving some input X , the sender sends some bits so that the receiver, using also the common randomness W , can simulate a single sample $Y \sim P_{Y|X}$. This point is so important that it is worth dwelling upon a little more. The sender's task **is not** to encode the conditional distribution $P_{Y|X}$ to the receiver. Of course, if they do encode $P_{Y|X}$, then the receiver can simulate a sample $Y \sim P_{Y|X}$, but encoding the whole distribution usually requires far more bits than just a message encoding enough information to draw one sample.

More formally, as was in the case for entropy coding algorithms (Section 41.2), REC algorithms comprise an encoder and a decoder, albeit slightly more involved. Concretely, let $\text{enc}_w : \mathcal{X} \rightarrow \{0, 1\}^*$ be a parametric function with parameter w , mapping from the source sample space $\mathcal{X}$ to finite-length bitstrings; we refer to enc_w as the encoder. Similarly, let $\text{dec}_w : \{0, 1\}^* \rightarrow \mathcal{Y}$ be a parametric function with parameter w , mapping from bitstrings to the output sample space $\mathcal{Y}$; we refer to dec_w as the decoder. The fundamental correctness requirement we impose on this pair of functions is as follows. Denoting the shared randomness as a random variable W , we require that for all $x \in \mathcal{X}$

$$\text{dec}_W(\text{enc}_W(x)) \sim P_{Y|X=x} \quad (42.17)$$

Before moving on, we emphasise two important points. First, the randomness in Equation 42.17 comes purely from the shared randomness W and not the source. Second, note that implicit in our requirement is the fact that the shared randomness is independent of the source X (it is difficult to imagine how it could be shared otherwise). Using dithered quantisation from the previous section as an example, the common randomness is $W = U \sim \text{Unif}(-1/2, 1/2)$, and the encoder outputs the entropy code corresponding to $K = \lfloor x + U \rfloor$, while the decoder recovers K and computes $K - U$.

Since we are interested in data compression, we are of course interested in the coding rate of the encoder as well, measured by $\mathbb{E}[\|\text{enc}_W(X)\|]$. We now demonstrate that the coding rate cannot be too small. For brevity, let $M = \text{enc}_W(X)$. Then,

$$\begin{aligned} \mathbb{E}[\|M\|] &\geq H[M | W] && \text{(Shannon's noiseless source coding theorem)} \\ &= H[M | W] - H[M | W, X] && (M \text{ is a function of } X \text{ and } W) \\ &= I[X; M | W] && \text{(definition of conditional mutual information)} \\ &= I[X; M | W] + I[X; W] && (X \text{ and } W \text{ are independent}) \\ &= I[X; (M, W)] && \text{(chain rule for mutual information)} \\ &\geq I[X; Y] && \text{(by Equation 42.17 and the data processing inequality)} \end{aligned}$$

Note that the above argument is valid for any encoder-decoder pair that satisfy Equation 42.17, and hence the mutual information $I[X; Y]$ provides a fundamental lower bound to our problem. On the other hand, we have already seen that dithered quantisation achieves this lower bound in the case of additive uniform perturbation; this finally motivates the definition of a relative entropy code.

Definition 42.5 (Relative Entropy Code). *For a source $X \sim P_X$ and conditional distribution $P_{Y|X}$, a relative entropy code is a triplet $(W, \text{enc}_w, \text{dec}_w)$ such that:*

1. W is a random variable independent of X , and is available both at the sender and the receiver's side.

⁷Recall that $I[X; Y] = D_{\text{KL}}(P_{X,Y} \| P_X \times P_Y)$.

2. The triplet satisfies Equation 42.17.
3. The coding rate of enc_w achieves at least

$$\mathbb{E}[\|\text{enc}_w(X)\|] = I[X; Y] + O(\log_2 I[X; Y]) \quad (42.18)$$

Given the discussion so far, the reader may be surprised that we allow for logarithmic rate overhead in Equation 42.18. However, unfortunately this additional logarithmic slack is necessary: it has been shown in the literature, that there exist examples where achieving $I[X; Y]$ within a constant additive term is not possible; see Braverman and Garg [203], Flamich et al. [534], Li and Gamal [1025] for example. Indeed, if we change our information measure from the mutual information $I[X; Y]$ to the *functional information* $F[X \rightarrow Y]$ defined recently by Hill et al. [747], we may get rid of the logarithmic slack. However, a proper discussion of the functional information is beyond the scope of the present discussion.

So far, we defined REC in the most general setting and demonstrated a fundamental lower bound to its efficiency. Next, we sketch a general technique for constructing a relative entropy code for any source $X \sim P_X$ and perturbation $P_{Y|X}$. The idea is to recognise that REC is a sampling problem, with the additional requirement that we be able to assign a code to the samples we simulate. Thus, it turns out that we can essentially use rejection sampling (Section 1.3) to construct a relative entropy code. Namely, there are two tricks we need:

1. We can use the marginal P_Y as the proposal distribution for rejection sampling.
2. We can set the proposal samples $Y_1, Y_2, \dots \sim P_Y$ as the common randomness $W = (Y_1, Y_2, \dots)$. In practice, the sender would use some PRNG seed s for generating these sequences, which they would then share with the decoder so that they can also generate the same random sequence.

Let $r_x(y) = \frac{dP_{Y|X=x}}{dP_Y}(y)$ be the density ratio of the posterior and the marginal, let $M_x \geq \|r_x\|_\infty$ be any upper bound on r_x and let $(U_1, U_2, \dots)$ be an infinite sequence of iid $\text{Unif}(0, 1)$ random variables. Then, recall that a rejection sampler finds the first proposed sample such that the acceptance criterion is satisfied; namely, they compute $K = \min\{k \in \mathbb{N} \mid U_k < r_x(Y_k)/M_x\}$, and we have that $Y_K \sim P_{Y|X=x}$. However, since we assumed that the sender and the receiver share all the Y_k s, it is enough for the sender to encode K , since the receiver can simply select $Y_K \sim P_{Y|X=x}$. It can be shown that we can encode K using $\mathbb{E}[\log K] + \log(\mathbb{E}[\log K] + 1) + 1$ bits. Using the fact that $K \sim \text{Geom}(1/M_x)$, by applying Jensen's inequality we get $\mathbb{E}[\log_2 K] \leq \log_2 \mathbb{E}[K] = \log_2 M_x$. Hence, using rejection sampling, we can construct a coding scheme with coding rate

$$\mathbb{E}[\|\text{enc}_w^{RS}(X)\|] \leq \mathbb{E}_X[\log_2 M_X] + \log_2(\mathbb{E}_X[\log_2 M_X] + 1) + 2 \text{ bits} \quad (42.19)$$

In the general case, the above coding rate can be well beyond the desired $I[X; Y] + O(\log_2 I[X; Y])$ cost that we initially wished for. However, Flamich [530] shows that when we require that the rejection sampler always terminate in expectation, that is, $\mathbb{E}[K] < C$ for some $C > 0$, we do indeed achieve the desired order:

$$\mathbb{E}[\|\text{enc}_w^{RS}(X)\|] \leq I[X; Y] + \log_2(I[X; Y] + 1) + \log_2 C + \log_2(\log_2 C + 1) + 2 \text{ bits} \quad (42.20)$$

However, this constant C can still be quite large in practice. Fortunately, more sophisticated algorithms, such as the Poisson functional representation/A* coding [533, 1025] and greedy Poisson rejection sampling [529] eliminate this inefficiency.

We also see another drawback of rejection sampling: its runtime. Namely, we saw that its expected runtime is $\mathbb{E}[K] = \mathbb{E}_X[M_X] \geq \mathbb{E}_X[\|r_X\|_\infty]$, which can be prohibitively large. Thus, the natural question is: could we do better? Unfortunately, in the general case, the answer is no: Goc and Flamich [642] show that the runtime of any general-purpose REC algorithm (that is, an algorithm applicable to any conditional distribution $P_{Y|X}$) is at least $\mathbb{E}_X[\|r_X\|_\infty]$. Upon a little thought, this result becomes less surprising: after all, relative entropy coding is a sampling problem, and it is well-known that sampling is a computationally hard problem in general [1072]. Therefore, fast REC algorithms, such as dithered quantisation (Section 42.2.1) and its variants [727, 909], branch-and-bound A* coding and GPRS [530] and PolarSim [1549] are all only applicable to encoding some restricted domain of perturbations. However, we note that it is still an open problem to find a fast algorithm for the practically most relevant perturbation: when $P_{Y|X}$ is multivariate Gaussian.

42.3 Applications of Relative Entropy Coding

In the previous sections, we have introduced the core concepts, theory, and algorithmic instances of REC. In this section, we reap the rewards of our hard work and look at some concrete applications of REC.

As we have already noted at the beginning of the chapter, the most straightforward benefit of REC over entropy coded quantisation, is that it has essentially no limitation on what probabilistic model we are using for the data. Thus, when it comes to lossy neural compression, we no longer need the approximations from Section 41.3.2 to train our models. As such, REC is a perfect match for improving the standard, autoencoder-based nonlinear transform coding methods. For example, we can combine any variational autoencoder (see Section 13.3.3) into a lossy data compression algorithm. For the interested reader, we recommend the paper by Flamich et al. [532] describing how this may be achieved in practice for variational autoencoders with Gaussian latent variables and the paper by Agustsson and Theis [11] using dithered quantisation (Section 42.2.1) for autoencoders with uniform latent variables. We note, that the approach of Agustsson and Theis [11] is very similar to the autoencoder-based nonlinear transform code introduced in Section 41.3.2. The main idea is that instead of switching to hard quantisation after training, they use dithered quantisation to encode samples from the additive uniform variational posterior. However, REC also enables us to construct compression algorithms using unconventional probabilistic models, such as Bayesian neural networks, which we illustrate in the rest of the section, and Gaussian diffusion models, which we leave as an exercise.

42.3.1 Compression with Bayesian implicit neural representations

Many data modalities can be viewed as continuous functions, including images, video, and many others. For instance, an image typically stored as a $W \times H$ array of RGB values can also be regarded as a function $f : [0, 1]^2 \rightarrow [0, 1]^3$ mapping spatial coordinates to (normalised) RGB colours. The discrete array provides samples $\{f(x_i, y_j)\}_{i \in [1:W], j \in [1:H]}$ on a grid; higher resolution corresponds to denser sampling of the same underlying function. Similarly, one can formalise a video clip as a mapping $f : [0, 1]^2 \times \mathbb{R}^+ \rightarrow [0, 1]^3$ (space-time to colour).

As a single data instance can be represented as a continuous function, we can fit a neural network to approximate this function, yielding a neural representation of that data point [1554]. As this representation is implicitly defined through a neural network, it is typically referred to as an *implicit neural representation (INR)*. In practice, this involves treating each data point as a dataset of coordinate-value pairs (c, v) , and *overfitting* a small multilayer perceptron (MLP) f to predict v from c .

This approach suggests that we could use INRs for lossy data compression: note that compressing an INR’s weights amounts to compressing the single data point it was fit to [465]; the data can be reconstructed by evaluating the INR at each coordinate. However, compressing the network weights directly is non-trivial: without specific treatment, the weights are continuous real values, and storing them at, e.g., 16-bit or 32-bit precision is highly inefficient. A common approach is to first quantise the weights to a lower-precision representation and entropy-code them. Unfortunately, INRs are highly sensitive to quantisation error. This contrasts with VAEs, where a powerful decoder network can learn to capture and compensate for the quantisation artefacts; in INRs, the learned MLP itself is the decoder, so any non-negligible quantisation error directly degrades reconstruction quality. Adding uniform noise during training is also suboptimal: different weights play different roles and have unequal sensitivities, while injecting the same bandwidth of uniform noise everywhere treats all parameters equally and hence will perform poorly.

The discussion above suggests a natural next step: instead of a single set of weights for the INR, we place a probability model over them. Concretely, we treat the INR weights as a random variable Y , specify a prior distribution P_Y , and, given a specific data point $X=x$, learn a posterior (by variational approximation) $Q_{Y|X=x}$ that captures which weights are plausible for this particular signal. This “Bayesian INR” provides us with a key advantage that aligns well with compression, as the uncertainty can adapt across layers and parameters, allowing noise to be injected more where the network is insensitive and less where it is fragile.

A key empirical observation for using Bayesian INRs for data compression is that the decoder does not need the full variational distribution: the reconstructions $\hat{x}$ we obtain using different weight samples from $Q_{Y|X=x}$ tend to be very similar. This observation immediately motivates using REC to encode only a single such realisation. Concretely, given a prior distribution P_Y and a variational posterior $Q_{Y|X=x}$ over the weights of an INR for a data point $X = x$, the sender can *encode a single stochastic realisation* $y \sim Q_{Y|X=x}$ with REC, and the receiver can recover the reconstructed data point using the INR with the weight sample y .

Now, we only need to consider how to obtain the variational posterior $Q_{Y|X=x}$ for a given data point, along with the prior distribution P_Y shared across all data points in one modality. We will explain these two parts respectively:

Inferring the posterior distribution $Q_{Y|X=x}$. Let $x = \{(c_i, v_i)\}_{i=1}^n$ be the coordinate-value pairs for a single data point x and let f_y be the INR with weights y . We train $Q_{Y|X=x}$ by balancing reconstruction quality and rate with the standard objective for variational inference, known as the negative β -Evidence Lower Bound, similarly

to other lossy neural compression techniques we already mentioned in Section 41.3.3:

$$\mathcal{L}_{\beta,x} = \underbrace{\sum_{(c,v) \in x} \mathbb{E}_{y \sim Q_{Y|X=x}} [\text{MSE}(v, f_y(c))]}_{\text{distortion (reconstruction error)}} + \underbrace{\beta \text{D}_{\text{KL}}(Q_{Y|X=x} \| P_Y)}_{\text{rate (approximate codelength)}}. \quad (42.21)$$

The parameter $\beta > 0$ controls the rate–distortion trade-off: larger β reduces bits at the cost of a larger compression error; smaller β yields better reconstructions but larger file size. We can see that, similar to the VAE case, this single objective both trains the INR and predicts its approximate codelength under REC.

Learning the prior P_Y . Data points within a domain tend to share structure. If we can learn a good prior P_Y from a *training set* that captures these commonalities in INR weights, we should expect better coding performance on the *test set*. But what makes a “good prior for compression more precisely? A good prior should minimise the *average* negative β -ELBO across training points, when the variational posterior for each training data point is optimised (see Chapter 12). This becomes a *nested optimisation task*. Therefore, a simple and effective recipe is the following coordinate descent-style algorithm:

1. **Fit posteriors for each training data point using the current prior.** For each x_i in the training set, minimise $\mathcal{L}_{\beta,x_i}$ w.r.t. $Q_{Y|X=x_i}$, parametrised as a tractable family (in practice, a diagonal Gaussian), where P_Y is given by the current prior.
2. **Fit the prior using the current posteriors for all training data.** For all x_i in the training set, minimise $\sum_i \mathcal{L}_{\beta,x_i}$ w.r.t. P_Y , where $\{Q_{Y|X=x_i}\}_i$ is given by the posteriors learned in the last step. In fact, this step adopts a closed-form solution if we choose P_Y to also be a diagonal Gaussian.

Putting everything above together, we obtain a lossy codec using REC and INRs. A few practical concerns remain: for example, what to do when $\text{D}_{\text{KL}}(Q_{Y|X=x} \| P_Y)$ is large; how to further reduce redundancy in INR weights; and how to handle very large data points. We refer readers interested in these details and more to the works of Guo et al. [688], Havasi et al. [715], He et al. [719].

Part IX

Applications

Chapter 43

Bayesian LLMs

Modern LLMs show outstanding performance in many scenarios and have taken the world by storm in just a few years. They are used for many tasks like complex reasoning, for example, math problem solving, theorem proving or question answering, as well as more traditional tasks like machine translation, summarization, or simply to improve the writing of an e-mail. And the list goes on and on. However, despite their massive success and widespread adoption, they are not without failure and many open problems persist, for example, calibration, uncertainty estimation and adaptation to new knowledge.

Bayesian methods are attractive for these problems already from a high-level point of view: the core of Bayesian learning, Bayes’ theorem, provides simple but powerful principles for model adaptation and prediction with model averaging should be more reliable as biases of single models are averaged out. When looking at Bayesian methods more in-depth one finds even further convincing arguments: Bayesian training can lead to flatter minima and model averaging allows for better treatment of uncertainty by decomposition of its sources.

Still, while many standard LLM techniques can have Bayesian interpretations [883], Bayesian methods are arguably not popular for LLMs yet. Indeed, conventional wisdom is that Bayesian methods are slow, do not work well, and are cumbersome to implement. In this chapter, we intend to show that this does not need to be true.

First, we show that existing LLMs can be improved by “making them more Bayesian” post-hoc in Section 43.1. Then, we show how LLMs can directly be trained using Bayesian methods in Section 43.2, how model averaging for LLMs can be done in a large output space (Section 43.3), Bayesian methods for uncertainty estimation (Section 43.4) and, finally, model merging from a Bayesian point of view (Section 43.5).

43.1 Bayesian fine-tuning of LLMs

Applying Bayesian inference to the weights of an LLM is challenging because of the model’s extremely high-dimensional parameter space (which might nowadays contain over one trillion parameters). This scale creates substantial difficulties for both posterior inference and posterior predictive inference:

1. During “training time” (i.e. posterior inference), approximate algorithms can suffer from slow mixing (e.g. MCMC-based approaches) or high gradient variance (e.g. VI-based approaches) due to the dimensionality of the posterior.
2. At “test time” (i.e. posterior predictive inference), Bayesian model averaging can be hard, since each sample from the posterior might require multiple gigabytes of disk storage, and repeatedly loading and unloading samples between the CPU and GPU might incur prohibitive overhead.

In addition to these practical challenges, the motivation for performing Bayesian inference over LLM weights is not immediately clear. Pre-trained LLMs are often well calibrated [1262], largely due to the enormous scale of their pre-training corpora. Consequently, recent works on Bayesian LLMs focus on the *fine-tuning* setting, in which models are trained on a small target dataset. The resulting fine-tuned models demonstrate significant overfitting, poor calibration, and a lack of uncertainty quantification. This is especially critical in safety-sensitive applications or when training data are scarce.

Parameter-efficient fine-tuning (PEFT) makes Bayesian fine-tuning a practical possibility. In particular, one widely adopted PEFT method is low-rank adaptation (LoRA) [788], which freezes the full pre-trained weight matrix $W \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$. Only the perturbation to the pre-trained weights, $\Delta W = \alpha \mathbf{B} \mathbf{A}$, is trained, where $\mathbf{B} \in \mathbb{R}^{D_{\text{out}} \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times D_{\text{in}}}$ are trainable low-rank matrices with a small rank $r \ll \min(D_{\text{in}}, D_{\text{out}})$ and constant scaling factor

$\alpha \in \mathbb{R}^+$. The forward pass of LoRA is

$$\mathbf{h} = (\mathbf{W} + \Delta\mathbf{W})\mathbf{x} = \mathbf{W}\mathbf{x} + \alpha\mathbf{B}\mathbf{A}\mathbf{x}. \quad (43.1)$$

Typically, $\mathbf{A}$ is initialized randomly, while $\mathbf{B}$ is initialized as a zero matrix. Thus, at initialization and start of fine-tuning, the forward pass is equivalent to that of the pre-trained model. During fine-tuning, the low-rank adapters are gradually learned. This reduces the number of trainable parameters from $D_{\text{in}}D_{\text{out}}$ to $r(D_{\text{in}} + D_{\text{out}})$ per module, and LoRA is implemented, for example, in the PEFT library [1129].

The advantages of performing Bayesian inference on $\mathbf{B}$ and $\mathbf{A}$, rather than $\mathbf{W}$, during fine-tuning are clear. The dimensions of $\mathbf{B}$ and $\mathbf{A}$ are much smaller than $\mathbf{W}$, making approximate inference engines easier to run. Bayesian model averaging becomes more feasible, since now each sample is a low-rank adapter rather than a full model, allowing many posterior samples to be loaded into GPU memory simultaneously.

Two early works in this direction are LoRA ensemble [1711] and Laplace-LoRA [1791], which apply Deep Ensemble and post-hoc linearized Laplace approximation on LoRA during fine-tuning, respectively.

43.1.1 Evaluation

Before describing the methodology, we first outline the evaluation setting for Bayesian fine-tuning. Note that in the Bayesian deep learning literature, a typical protocol involves training a model from scratch and then evaluating metrics such as likelihood, predictive entropy, or expected calibration error (ECE) on out-of-distribution (OOD) data. The usual promise of Bayesian deep learning approaches is that models will exhibit higher uncertainty on OOD data, whereas the standard point-estimation approach shows degraded accuracy and overconfident predictions.

Similarly, in Bayesian fine-tuning, we adapt the LLM to small multiple-choice QA datasets, such as CommonsenseQA [1584], MMLU [734], or AI2 Reasoning Challenge [312]. The LLM is trained to classify among the answer options given a question prompt.

For evaluation, we test the Bayesian fine-tuned LLM on the held-out evaluation split, where we expect the model to show high accuracy, low ECE, and low negative log-likelihood (NLL). Compared to a point-estimate LoRA, we anticipate fewer confident but incorrect predictions. For OOD evaluation, we choose datasets that differ significantly from the training data, e.g., QA questions in other topics or data subcategories, where we expect the Bayesian approaches to LoRA to show higher uncertainty, allowing us to detect the OOD samples.

43.1.2 LoRA ensemble

To address overconfidence and poor calibration, LoRA ensemble [1711] trains multiple LoRA adapters independently using different initializations. Then, at test time, we average the predictive distributions of the M ensemble members (Section IV). Because storing and loading multiple full copies of LLMs onto the GPU at test time is prohibitively expensive, all ensemble members share the same pre-trained model, and we only fine-tune the LoRA adapters. Thus, the storage and model-loading overhead is nearly the same as when using the original model. A single LLaMA 13B model at 16-bit precision requires 25 GB on disk and takes approximately 6 seconds to load, while a LoRA adapter of rank 8 only takes 30 MB on disk and takes 0.1 seconds to load. This highlights the practical advantage of LoRA ensemble over storing multiple complete LLMs in memory. Importantly, in LoRA ensemble, the large pre-trained model is loaded only once.

The idea is simple: we average the predictive distributions of each individual LoRA:

$$p_{\text{ens}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{M} \sum_{m=1}^M p(\mathbf{y} \mid \mathbf{x}; \mathbf{W} + \alpha\mathbf{B}_m\mathbf{A}_m). \quad (43.2)$$

The different adapters $\mathbf{A}_m$ and $\mathbf{B}_m$ provide the necessary diversity, together with the randomness from the data shuffling (i.e. SGD noise). LoRA ensemble gives considerable improvements in accuracy and calibration, compared to single-component LoRA fine-tuning, last-layer fine-tuning, Monte Carlo dropout, and few-shot prompting (where a few examples of questions and answers are appended in front of the prompt). LoRA ensemble can also be combined with standard regularization approaches, such as weight decay, KL regularization, and early stopping, which keep the output distribution close to that of the pre-trained model. This can generally further improve the predictive accuracy and uncertainty quantification. Furthermore, using larger M improves all metrics, yet the gains become less substantial as M increases (the main experiments use $M = 5$). Lastly, LoRA ensembling is also helpful in OOD settings, leading to lower NLL and ECE. However, it does not necessarily improve OOD detection performance.

43.1.3 Laplace-LoRA

Laplace-LoRA [1791] addresses the calibration of the fine-tuned model by applying a Laplace approximation (Section IV) to the posterior over the LoRA adapters. The key advantage of the method is that the Laplace approximation is used post-hoc, therefore, both pre-training and fine-tuning remain unchanged. Computing Laplace approximation requires computing a $P \times P$ curvature matrix, where P is the number of parameters, which for a full model is infeasible. Laplace-LoRA mitigates this by considering only the LoRA weights. For LLaMA 2 7B, this corresponds to only approximately 6 million parameters (around 0.1% of the total model parameters). This scale remains impractical and can be further addressed by a structured approximation, a block-diagonal Kronecker-factored approximation (KFAC). Each adapter $\Delta W = \mathbf{B}\mathbf{A}$ is treated as two separate linear layers, with weights $\mathbf{A}$ and $\mathbf{B}$. However, for $\mathbf{A} \in \mathbb{R}^{r \times D_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{D_{\text{out}} \times r}$, one Kronecker factor is small ($r \times r$), whereas the other is large, with dimensions $(D_{\text{in}} \times D_{\text{in}})$ for $\mathbf{A}$ or $(D_{\text{out}} \times D_{\text{out}})$ for $\mathbf{B}$. Laplace-LoRA computes a low-rank approximation of the large Kronecker factor. Additionally, the method allows the prior precision λ to be optimized using marginal likelihood, and does not require a separate held-out validation set which is particularly beneficial for very small datasets. All in all, the resulting posterior approximation is:

$$p(\mathbf{A}, \mathbf{B} \mid \mathcal{D}) \approx \mathcal{N}\left(\begin{bmatrix} \text{vec}(\mathbf{A}_*) \\ \text{vec}(\mathbf{B}_*) \end{bmatrix}, \Psi_{\mathbf{A}, \mathbf{B}}^{-1}\right) \quad (43.3)$$

Crucially, to ensure an efficient implementation, the method uses an incremental low-rank SVD to estimate the low-rank components, without ever computing factors of size $(D_{\text{in}} \times D_{\text{in}})$ or $(D_{\text{out}} \times D_{\text{out}})$. Large values of negative log-likelihood, as well as, expected calibration error are reduced significantly, compared with MAP and other approaches such as Monte Carlo dropout, checkpoint ensemble (ensembling most recent checkpoints) or temperature scaling. Laplace-LoRA is compatible with regularized approaches, as it can be applied to any LoRA-based method that generates fine-tuned weights. Lastly, Laplace-LoRA proves to be useful in evaluations under both smaller and larger distribution shifts, with regard to NLL and ECE.

43.2 Bayesian LLM Training with Variational Learning

We discuss training LLMs directly using Bayesian methods. While conventional LLM training predominantly learns point estimates $\boldsymbol{\theta} \in \mathbb{R}^P$ of one specific LLM, we instead aim to obtain a distribution over them which also quantifies uncertainty about the parameter values. While this is also possible by using conventional point estimation followed by Laplace’s method, as we have seen in the previous section, we here provide arguments why training a distribution directly can be beneficial. We will show methods that work even at large scales, contrary to the popular belief that Bayesian methods are not scalable to large problems like language modeling. The methods presented here use the framework of variational learning and the natural-gradient-based algorithms which we have introduced formally in Section 15.1. In the following, we will briefly review the formal introduction and then go over to practical aspects of scaling variational learning to LLMs, where we first discuss training, then prediction, and finally adaptation to new data.

43.2.1 Variational Learning for Language Models

Modern LLM training consists of a sequence of multiple stages. The first stage generally uses a language modeling loss,

$$\ell(\mathcal{D}, \boldsymbol{\theta}) = - \sum_{\mathbf{y} \in \mathcal{D}} \log p(\mathbf{y} \mid \boldsymbol{\theta}) \quad (43.4)$$

$$= - \sum_{\mathbf{y} \in \mathcal{D}} \log \left[\prod_{t=1}^{|\mathbf{y}|} p(y_t \mid y_1, \dots, y_{t-1}, \boldsymbol{\theta}) \right], \quad (43.5)$$

that aims to maximize the likelihood of tokens in a large collection of texts $\mathcal{D}$, where $|\mathcal{D}|$ can be on the order of trillions of tokens. The composition of $\mathcal{D}$ is then often changed in later stages of training, for example, to include more math and code data in mid-training, or to include data used for safety tuning in post-training, where the model is fine-tuned for specific capabilities [967, 1139, 1700]. These capabilities can also include input-specific data, denoted by $\mathbf{x}$. These inputs often also contain human instructions given to the model [213]. In addition to the data composition, the training criterion often changes, as well. For example, to improve the reasoning capabilities of models, reinforcement learning is often used, where the loss is not always a likelihood anymore but

might be a reward assigned by some heuristic function or a learned model [1273]. Still, in any case the optimizer is usually not switched but only the loss function and gradient estimator.

When training with Bayesian methods we would like to have the same property: the loss function might change but ideally the overall framework and optimization algorithm should stay the same to ease implementation. Variational learning provides such a framework and aims to approximate a posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ by a (generalized) posterior:

$$q_*(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{i=1}^N \exp\{-\ell(\mathcal{D}_i, \boldsymbol{\theta})\}. \quad (43.6)$$

Here p_0 is a prior and $\ell(\mathcal{D}_i, \boldsymbol{\theta})$ can be any loss, though we usually assume it is at least once differentiable. The posterior is learned using the following variational objective:

$$q_*(\boldsymbol{\theta}) = \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} \mathbb{E}_q \left[\sum_{i=1}^N \ell(\mathcal{D}_i, \boldsymbol{\theta}) \right] + \mathbb{D}_{\text{KL}}(q \parallel p_0), \quad (43.7)$$

where $\mathcal{Q}$ is a class of distributions, for example, Gaussian distributions to make optimization tractable. For language modeling the objective can then also be written as

$$q_*(\boldsymbol{\theta}) = \arg \min_{q(\boldsymbol{\theta}) \in \mathcal{Q}} -\mathbb{E}_q \left[\sum_{\mathbf{y} \in \mathcal{D}} \log \left[\prod_{t=1}^{|\mathbf{y}|} p(y_t \mid y_1, \dots, y_{t-1}, \boldsymbol{\theta}) \right] \right] + \mathbb{D}_{\text{KL}}(q \parallel p_0) \quad (43.8)$$

but we are now not bound to optimizing likelihoods anymore and can easily change the loss function, also during optimization, for the different stages of language model training. Oftentimes, the loss is also normalized by sequence length to improve training stability.

43.2.2 IVON for LLMs

Algorithm 35: Improved Variational Online Newton (IVON). Main differences to Adam in grey. Note that we overload $/$ for notational convenience.

Inputs : learning rates $\{\rho_t\}$; gradient momentum $\beta_1 \in [0, 1]$; Hessian momentum $\beta_2 \in [0, 1]$; weight decay $\delta > 0$; scaling parameter $\lambda > 0$; Hessian initialization $h_0 > 0$, clip radius $\xi > 0$

Outputs: mean parameter $\mathbf{m}_{t+1}$ and variance σ_{t+1}^2

initialize $\mathbf{m} \leftarrow$ (NN weight init), $\mathbf{h} \leftarrow h_0 \cdot \mathbf{I}$, $\boldsymbol{\sigma}^2 \leftarrow 1/(\lambda(\mathbf{h} + \delta))$, $\mathbf{g} \leftarrow 0$, $t \leftarrow 1$, and $\alpha_t \leftarrow (h_0 + \delta)\alpha_t$ for all t

repeat

$\hat{\mathbf{g}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}_i} \ell_i(\boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t \sim q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{m}_t, \text{diag}(\boldsymbol{\sigma}_t^2))$

$\hat{\mathbf{h}} \leftarrow \hat{\mathbf{g}} \cdot (\boldsymbol{\theta}_t - \mathbf{m}_t) / \sigma_t^2$

$\mathbf{g} \leftarrow \beta_1 \mathbf{g} + (1 - \beta_1) \hat{\mathbf{g}}$

$\mathbf{h} \leftarrow \beta_2 \mathbf{h}_t + (1 - \beta_2) \hat{\mathbf{h}} + \frac{1}{2} (1 - \beta_2)^2 (\mathbf{h}_t - \hat{\mathbf{h}})^2 / (\mathbf{h}_t + \delta)$

$\bar{\mathbf{g}} \leftarrow \mathbf{g} / (1 - \beta_1^t)$

$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \alpha_t \text{clip}((\bar{\mathbf{g}} + \delta \mathbf{m}_t) / (\mathbf{h} + \delta), \xi)$

$\sigma_{t+1}^2 \leftarrow 1/(\lambda(\mathbf{h} + \delta))$

until convergence

return $\mathbf{m}_{t+1}, \sigma_{t+1}^2$

Interestingly, practical algorithms that optimize Equation 43.7 take a very similar form to optimizers that are already known to work well for deep learning, like Adam [894], but now calculate natural gradients to quantify steepest descent in terms of distributions and not point estimates.¹ One of these optimizers is the IVON optimizer shown in Algorithm 35, where the main differences to Adam are highlighted in grey. IVON is formally derived in Section 15.3.2.

The first difference is that IVON learns a Gaussian distribution over model parameters instead of a point estimate. Usually a diagonal covariance is modeled for LLMs but a non-diagonal covariance can also be learned using a recent extension called EVON [1184]. Both of these choices have a similar computational and memory

¹This is explained in more detail in Section 15.1.

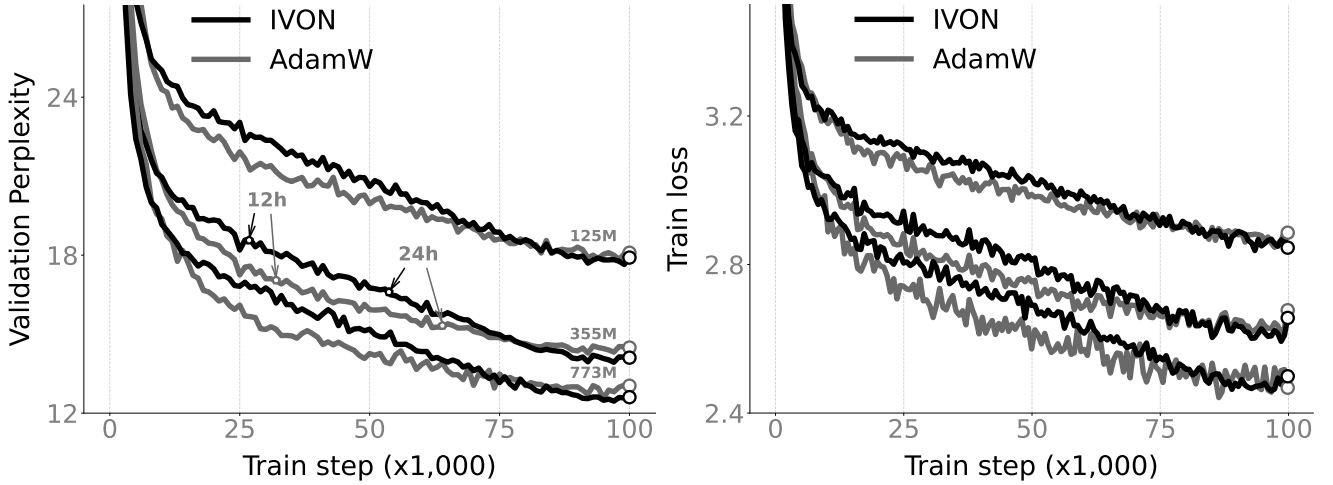


Figure 43.1: Results when pretraining different GPT-2 models using IVON from scratch compared to AdamW. The figure shows validation perplexity (left) and train loss (right). In both cases IVON can improve over AdamW at similar cost. Figure adapted from Shen et al. [1510] with permission.

overhead as AdamW. For example, in the experiments shown in Figure 43.1 IVON has an overhead of ca. 1.2x. The variance in IVON is defined using the inverse of a diagonal Hessian approximation and therefore the update is also preconditioned by the inverse of a diagonal Hessian approximation, instead of a square root second moment estimate in AdamW. In IVON, the diagonal Hessian is estimated by using the reparameterization trick instead of a square of minibatch gradients but the estimator could be switched in principle. Finally, the Hessian estimate is ensured to stay positive by the additional $(1 - \beta_2)^2(\cdot)$ term in its update.

One benefit of the reparameterization trick is that it requires sampling parameters at each iteration, which is equivalent to adding learned noise to the current model. As a consequence, IVON constantly explores local neighborhoods of the current mean during optimization which can keep it away from sharp minima and push it to flat minima [620] which have been associated with, among others, improved generalization [544].

With these changes IVON has been successfully used to train language models up to 773M parameters following the GPT-2 architecture in Shen et al. [1510]. The results are summarized in Figure 43.1, where IVON even gets better validation perplexity (evaluated at the mean of the distribution) at the end of training. Consistent with the language modeling loss, perplexity is defined as follows for a corpus $\mathcal{D}$:

$$\text{PPL}(\mathcal{D}, \theta) := \exp \left(-\frac{1}{\sum_{\mathbf{y} \in \mathcal{D}} |\mathbf{y}|} \sum_{\mathbf{y} \in \mathcal{D}} \sum_{t=1}^{|\mathbf{y}|} \log p(y_t | \mathbf{y}_{<t}, \theta) \right), \quad (43.9)$$

where the shorthand $\mathbf{y}_{<t}$ describes $\mathbf{y}$ up until the $(t-1)$ -th token. The authors also show that sampling models from the posterior and averaging their predictions can improve perplexity. In order to make IVON work for Transformer LLMs, the preconditioned gradient is clipped to a value ξ which can be set, for example, to 10^{-4} . In general, hyperparameters differ quite a bit from AdamW and are important to make the method work. Typically, the gradient momentum $\beta_1 = 0.9$ is similar but β_2 is often chosen to be much larger, for example, 0.99999 or 0.9999 which stabilizes training as the reparameterization-trick-based Hessian estimator can be noisy. The noise can also be reduced by averaging the gradient estimates over multiple Monte-Carlo estimates for each step instead of using one sample. For LLMs it is also important to tune the Hessian initialization h_0 and the effective sample size λ , because both are directly related to the amount of noise injected during optimization. Larger values can stabilize training and therefore it can be good practice to treat it as a hyperparameter rather than using the dataset size directly, especially when fine-tuning. Typically, values like $\lambda = 10^7$ and $h_0 \in [0.01, 1.0]$ can work well but other combinations are also possible and it is up to the practitioner to balance exploration via noise injection and optimization stability. The view as an inverse temperature parameter is justified by the authors of Shen et al. [1510] by rewriting the variational objective as:

$$q_*(\theta) = \arg \min_{q(\theta) \in \mathcal{Q}} \left[\sum_{i=1}^N \bar{\ell}(\mathcal{D}_i, \theta) \right] - \frac{1}{\lambda} \mathcal{H}(q). \quad (43.10)$$

Here, the rewriting is due to choosing a regularizer $r(\boldsymbol{\theta})$ such that the prior is defined as $p_0(\boldsymbol{\theta}) \propto \exp(-r(\boldsymbol{\theta}))$ [883]. The regularizer is subsumed in the loss which is averaged over all data examples and denoted with $\bar{\ell}$. The rewriting shows that smaller λ prefer a larger entropy in q which increases diversity in models but can decrease data-fit.

43.2.3 LoRA Fine-tuning with IVON

Recently, IVON has also been used to improve LoRA fine-tuning, where the distribution is not learned over the whole model but rather only over the LoRA parameters [326]. Still, many of the benefits of Bayesian Deep Learning are retained and, in particular, the method can achieve improvements in both accuracy and calibration at the same time. The authors also introduce a pruning method based on the posterior which they call uncertainty-guided pruning. There, the model parameters are pruned via their posterior variance

$$\sigma_i^{-2} = \lambda(\mathbf{h} + \delta), \quad (43.11)$$

where $\mathbf{h}$ is the Hessian estimate from IVON and δ is a standard weight-decay. The diagonal Hessian can be seen as an estimate of (inverse) uncertainty over the parameter: a larger Hessian means less uncertainty as the variance is small and the posterior at this point is concentrated for the given parameter and vice versa. The most uncertain parameters can be pruned to improve efficiency and performance.

The parameter uncertainty is also useful because it allows us to sample models by adding noise to the mean proportional to the variance. Then, we can generate language by taking an expectation over predictions across (samples from) the approximate posterior and also, again, adjust the (inverse) temperature to make predictions more or less diverse. In the following sections we will deal with this in greater detail and outline practical language generation methods based on Bayesian Deep Learning.

43.3 Language Generation with Predictive Posteriors

Language generation is challenging, because the output space is huge and many outputs can be equally valid for a given input. Which of these valid outputs is preferred often depends on user preference. Bayesian methods can deal with such uncertainty via predictive posteriors, where predictions are averaged across models, but applying them requires dealing with nuances, starting from simply defining predictive posteriors to estimating expected risks that can be used to account for preference. Here, we will first review common language generation strategies and then derive generalized Bayesian strategies.

Suppose we are given some input $\mathbf{x} \in \mathcal{V}^*$, where the Kleene closure $*$ constructs all possible finite-length outputs from a vocabulary $\mathcal{V}$ that may be based on, for example, byte-pair encoding (BPE) [1490]. We are interested in finding an output $\mathbf{y} \in \mathcal{V}^*$ constructed of a sequence of tokens $y_1, y_2, \dots, y_{|\mathbf{y}|}$. As discussed before, LLMs usually define the probability of $\mathbf{y}$ autoregressively by taking the product of per-token probabilities:

$$p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t | \mathbf{y}_{<t}, \mathbf{x}, \boldsymbol{\theta}). \quad (43.12)$$

One common strategy is to sample $y_t \sim p(\cdot | \mathbf{y}_{<t}, \mathbf{x}, \boldsymbol{\theta})$ from the model until a stopping criterion, for example, a special end-of-sequence symbol. However, due to the common softmax activation, which produces dense probabilities, and uncertainty over the correct y_t an incorrect token might always be sampled with some probability, for example, an incorrect fact. One might think that, perhaps, we could use the most likely token $y_t^* = \max_{y_t \in \mathcal{V}} p(\cdot | \mathbf{y}_{<t}, \mathbf{x}, \boldsymbol{\theta})$ instead but this greedy decoding assumes a good point estimate on the one hand, and on the other, in general, does not find the most likely *sequence*

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathcal{V}^*} p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta}). \quad (43.13)$$

While using a posterior $q(\boldsymbol{\theta})$ should intuitively give us better probability estimates due to averaging, we first again need to decide which probabilities should be considered. We could either define [1125]:

$$p^{\text{tok}}(\mathbf{y} | \mathbf{x}) := \prod_{t=1}^T \mathbb{E}_q [p(y_t | \mathbf{y}_{<t}, \mathbf{x}, \boldsymbol{\theta})] \quad (43.14)$$

or similarly define

$$p^{\text{seq}}(\mathbf{y} | \mathbf{x}) := \mathbb{E}_q [p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta})] = \mathbb{E}_q \left[\prod_{t=1}^T p(y_t | \mathbf{y}_{<t}, \mathbf{x}, \boldsymbol{\theta}) \right]. \quad (43.15)$$

Method	IWSLT17 En-De			WMT18 Tr-En		
	BLEU	COMET	LaBSE	BLEU	COMET	LaBSE
IVON@Mean	19.73	76.60	83.51	15.27	78.44	77.12
Pred. Posterior - Equation 43.22	20.89	77.42	84.01	15.66	79.01	77.79
Pred. Posterior - Equation 43.23	21.08	77.63	83.96	15.46	78.84	77.35

Table 43.1: Results when using the IVON mean and posterior for language generation with uncertainty-aware MBR on two machine translation tasks. The loss used is the negative BERTScore and all methods use 40 output samples in total. Taken from Daheim et al. [354].

However, both will not give the same probabilities for a given $\mathbf{y}$ in general. In the end, this is a choice of the practitioner that also depends on implementation and storage requirements.

When attempting to find likely sequences, there is also mismatch between training and prediction: when using general losses during training, we might also want to consider them during generation, and vice versa. For example, in machine translation we would rather want an output with high quality than high probability [558] and the two do not necessarily coincide. As a consequence, estimates of quality have also been used as training criteria [517, 723]. In the following, we present generalized Bayesian language generation methods that can effectively use both posterior and losses.

43.3.1 Uncertainty-Aware Minimum Bayes Risk Decoding with Posteriors

The use of losses in language generation is not a new concept. In fact, Minimum Bayes Risk (MBR) decoding has long been used for sequence generation [947] from a point estimate $\boldsymbol{\theta} \in \mathbb{R}^p$ using the expected risk:

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in \mathcal{V}^*} \sum_{\mathbf{y}' \in \mathcal{V}^*} p(\mathbf{y}' | \mathbf{x}, \boldsymbol{\theta}) \ell(\mathbf{y}, \mathbf{y}'), \quad (43.16)$$

where $\ell : \mathcal{V}^* \times \mathcal{V}^* \rightarrow \mathbb{R}_+$ is a loss function that compares two strings. Interestingly, this algorithm generalizes many well-known strategies already. For example, we could derive Equation 43.13 by choosing a 0 – 1-loss. Another example is choosing $\ell(\mathbf{y}, \mathbf{y}') = 1\{f(\mathbf{y}) \leq f(\mathbf{y}')\}$ for some $f : \mathcal{V}^* \rightarrow \mathbb{R}_+$, for example, a translation quality estimate. This recovers a ranking according to f whose approximations have recently been popularized under the term quality-aware decoding [516].

By averaging such expected risks across a posterior $q(\boldsymbol{\theta})$ we then arrive at the generalized Bayesian language generation method from Daheim et al. [354]:

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in \mathcal{V}^*} \mathbb{E}_q \left[\sum_{\mathbf{y}' \in \mathcal{V}^*} p(\mathbf{y}' | \mathbf{x}, \boldsymbol{\theta}) \ell(\mathbf{y}, \mathbf{y}') \right] \quad (43.17)$$

$$= \arg \min_{\mathbf{y} \in \mathcal{V}^*} \sum_{\mathbf{y}' \in \mathcal{V}^*} \mathbb{E}_q [p(\mathbf{y}' | \mathbf{x}, \boldsymbol{\theta}) \ell(\mathbf{y}, \mathbf{y}')] \quad (43.18)$$

$$= \arg \min_{\mathbf{y} \in \mathcal{V}^*} \sum_{\mathbf{y}' \in \mathcal{V}^*} p^{\text{seq}}(\mathbf{y}' | \mathbf{x}) \ell(\mathbf{y}, \mathbf{y}'), \quad (43.19)$$

where the switch of expectations is valid, because the size of $\mathcal{V}^*$ is countably-infinite, ℓ is non-negative, and sequence-level probabilities over $\mathbf{y}$ are used. By defining the loss,

$$\ell(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) = \sum_{\mathbf{y}' \in \mathcal{V}^*} p(\mathbf{y}' | \mathbf{x}, \boldsymbol{\theta}) \ell(\mathbf{y}, \mathbf{y}'), \quad (43.20)$$

we can also make the correspondence to Equation 43.7 more explicit by noting that,

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in \mathcal{V}^*} \mathbb{E}_q [\ell(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})], \quad (43.21)$$

which clearly shows how we can optimize the same objective during training and prediction.²

In practice, both expectations over q and $\mathcal{V}^*$ are intractable to compute. Therefore, Daheim et al. [354] use Monte-Carlo estimation which first uses the following two steps:

²Note that in Equation 43.7 we optimize in $\mathcal{Q}$ and the regularizer over $q(\boldsymbol{\theta})$ is constant during generation, where $q(\boldsymbol{\theta})$ is fixed.

1. Sample a set $\mathcal{M} = \{\theta_i \mid \theta_i \sim q(\theta)\}_{i=1}^{|\mathcal{M}|}$ of models from the posterior q .
2. For each of them, sample a hypothesis set $\mathcal{H}_{\theta_i} = \{\mathbf{y}^{(j)} \mid \mathbf{y}^{(j)} \sim p(\cdot \mid \mathbf{x}, \theta_i)\}_{j=1}^{|\mathcal{H}|}$.

The hypothesis sets are then used to estimate Equation 43.17 by using:

$$\hat{\mathbf{y}}^* = \arg \min_{\mathbf{y}' \in \mathcal{H}_{\mathcal{M}}} \sum_{\mathbf{y} \in \mathcal{H}_{\mathcal{M}}} \ell(\mathbf{y}, \mathbf{y}') \quad (43.22)$$

or Equation 43.18 with

$$\hat{\mathbf{y}}^* = \arg \min_{\mathbf{y}' \in \mathcal{H}_{\mathcal{M}}} \sum_{\theta \in \mathcal{M}} \sum_{\mathbf{y} \in \mathcal{H}_{\theta}} \ell(\mathbf{y}, \mathbf{y}'). \quad (43.23)$$

Here $\mathcal{H}_{\mathcal{M}} := \cup_{\theta \in \mathcal{M}} \mathcal{H}_{\theta}$ is a collection of all sampled outputs from all models, where counts are kept to maintain an unbiased estimator. Intuitively, one uses the probability in Equation 43.15 for risk calculation and the other averages per-model risks which is also easier to parallelize. While both may give different estimates in practice, they are still unbiased and will converge to the same risk in the limit and its up to the practitioner to choose. It is also possible to derive a similar decision rule for token-level averaging by replacing the probability in Equation 43.19 with a token-level averaging.

In Daheim et al. [354], the authors find that both methods perform better than using the mean of an approximate posterior learned with IVON. We show an excerpt of their results on two machine translation tasks, namely the IWSLT17 [251] en-de and WMT18 [182] tr-en task in Table 43.1. The model is GEMMA-2B [604] fine-tuned using IVON-LoRA [326]. The loss function is the negative of the BERTScore [1857] between two utterances and each method uses 40 samples in total which means that 40 samples were taken from the mean and 10 for each of the 4 sampled models from the posterior. The results show that, overall, the predictive posterior helps when generating language. This is also shown in Cong et al. [326], who show that scaling the number of models and samples in MBR can be an effective method for test-time compute scaling, for example, for math reasoning with LLMs. In addition, the expected risk is useful for selective prediction which we will discuss next.

43.3.2 Selective Prediction using Expected Risk

When generating language, there can also be cases where it is not possible to generate any useful output. This is covered by selective prediction, as discussed in Chapter 44. There, the idea is to refrain from presenting an output to the user if the output is expected to not be useful. For example, in machine translation we might encounter inputs that are in a language that is not supported by our translation model or in a question answering setting answering might require knowledge that is not present in the model.

The expected risk defined in the previous section can be a useful way of putting this idea into practice, as we might want to refrain from answering when risk is high. Then, it makes sense to define an expected risk across all outputs as follows:

$$\text{risk}(\mathbf{x}, q) = \mathbb{E}_{\mathbf{y}} \mathbb{E}_q \left[\underbrace{\sum_{\mathbf{y}' \in \mathcal{V}^*} p(\mathbf{y}' \mid \mathbf{x}, \theta) \ell(\mathbf{y}, \mathbf{y}')}_{:= \text{risk}(\mathbf{y}, \mathbf{x}, q)} \right]. \quad (43.24)$$

The risk quantifies the expected risk of *all* possible outputs $\mathbf{y} \in \mathcal{V}^*$ across the posterior q which we obtain by comparing it to each other output $\mathbf{y}' \in \mathcal{V}^*$.

Again, this is clearly intractable to calculate in practice and, using notation from previous subsections, we can therefore construct an estimator

$$\text{risk}(\mathbf{x}, q) \approx \frac{1}{|\mathcal{M}|} \frac{1}{|\mathcal{H}_{\theta_i}|^2} \sum_{\theta_i \in \mathcal{M}} \sum_{\mathbf{y} \in \mathcal{H}_{\theta_i}} \sum_{\mathbf{y}' \in \mathcal{H}_{\theta_i}} \ell(\mathbf{y}, \mathbf{y}'). \quad (43.25)$$

The authors of Daheim et al. [354] show that this can be useful for selective prediction, especially when the $\mathcal{H}$ are created with ancestral sampling from a diverse set of models. In IVON, this can also be achieved by decreasing λ during prediction.

43.4 Uncertainty Estimation for Autoregressive LLMs

Bayesian methods are also attractive because they can improve predictive uncertainty estimation and enable distinguishing different sources of uncertainty. Most commonly, a distinction is made between uncertainty stemming

from a lack of knowledge, often called epistemic uncertainty, and uncertainty stemming from randomness in the data itself, often called aleatoric uncertainty [415, 804]. While the former is assumed to be reducible by observing (and training on) more data, the latter is generally thought of as irreducible, because it reflects inherent uncertainty. A simple canonical example is (unbiased) coin tossing, where a model trained on few coin tosses might not equally assign probability to heads and tails, but observing a sufficient amount of data will eventually lead to convergence to a uniform distribution. In LLMs randomness is clearly much more nuanced. For example, it can stem from many outputs being equally valid, speaker perspectives, data collection, and so on. Then, whether certain sources are reducible may even be a choice. We refer the reader to Baan et al. [70] for a further discussion and instead discuss information-theoretic measures for disentangling uncertainties next.

There, it is usually assumed that total uncertainty when predicting any $\mathbf{y}$ given $\mathbf{x}$ is captured by the entropy of the predictive distribution. That is, the entropy captures both reducible and irreducible uncertainty. In order to decompose it, we will use the mutual information between $\mathbf{y}$ and the network weights $\boldsymbol{\theta}$ which tells us how much information about $\boldsymbol{\theta}$ we can gain by observing $\mathbf{y}$ and is given by subtracting a measure of aleatoric uncertainty from total uncertainty [804, 1125]:

$$\mathcal{I}[\mathbf{y}, \boldsymbol{\theta} \mid \mathbf{x}] = \underbrace{\mathcal{H}[p^{\text{seq}}(\mathbf{y} \mid \mathbf{x})]}_{\text{Tot. Uncertainty}} - \underbrace{\mathbb{E}_q[\mathcal{H}[p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})]]}_{\text{Aleat. Uncertainty}}. \quad (43.26)$$

However, as so often, calculating these quantities exactly is not feasible and instead approximations have to be used, such as Monte-Carlo approximations. These can also be tricky for sequence-level model averaging and as a consequence token-level model averaging may be used, too, but as we discussed previously is not guaranteed to be consistent with sequence-level modeling. We refer the reader to Malinin and Gales [1125] for an in-depth discussion of such estimators and, in particular a comparison of token- and sequence-level modeling.

We also note that these measures are clearly related to the expected risk formulations we have seen in the previous section via a suitable loss function. Using these intuitions, Vashurin et al. [1668] therefore combine information-theoretic measures and general loss functions for improved uncertainty estimation and compare this approach to a variety of baselines, especially when estimating uncertainty given only a single output sample.

Monte-Carlo estimators are also challenging because for LLMs it can be unlikely that the same sequence will be sampled multiple times, even though many sequences share the same meaning. Therefore, Kuhn et al. [942] group sequences with the same meaning before calculating entropy which they call *semantic uncertainty* and which has shown state-of-the-art performance in many settings. Let us denote a cluster of sequences with the same meaning as c . Then, we can define the probability of c for given $\boldsymbol{\theta}$ as

$$p(c \mid \mathbf{x}, \boldsymbol{\theta}) := \sum_{\mathbf{y} \in c} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}). \quad (43.27)$$

For a set of meanings C , semantic uncertainty is then given by the redefined entropy

$$SE[p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})] := - \sum_{c \in C} p(c \mid \mathbf{x}, \boldsymbol{\theta}) \log p(c \mid \mathbf{x}, \boldsymbol{\theta}). \quad (43.28)$$

Although extensions to Bayesian modeling with predictive posteriors are not used in Kuhn et al. [942], these can be easily obtained using the definitions of total, aleatoric, and epistemic uncertainty outlined in this section. In fact, under this lens Equation 43.28 may already be seen as an approximation to Bayesian methods by approximating the expectation over an approximate posterior at the mean. This is often practical for LLMs which might only be available as a point estimate.

43.5 Bayesian Model Merging for LLMs

Recently, model merging has gained attention as a way to combine the information from multiple models $\boldsymbol{\theta}_t$ that were trained separately on T different $\mathcal{D}_t$, for example, by a simple weighted-averaging of their parameters using scaling factors $\alpha_t > 0$:

$$\hat{\boldsymbol{\theta}}_{1:T} = \sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t, \quad (43.29)$$

where the α_t usually sum to 1. This can be seen as approximating a model $\boldsymbol{\theta}_{1:T}$ that is trained on all $\mathcal{D}_t$ altogether with a similar weighting [353], for example by minimizing an empirical risk with a weight decay:

$$\boldsymbol{\theta}_{1:T} = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^p} \sum_{t=1}^T \alpha_t \ell(\mathcal{D}_t, \boldsymbol{\theta}) + \frac{1}{2} \|\boldsymbol{\theta}\|_2^2. \quad (43.30)$$

We largely follow Maldonado et al. [1124] to connect such methods to Bayesian methods. To a Bayesian, the model merging in Equation 43.29 might seem familiar as the averaging of models should intuitively correspond to a multiplication of likelihoods which can be used to construct a posterior. Using similar weights α_t we can write a posterior $p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T})$ by multiplying likelihoods $p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_t)/p_0(\boldsymbol{\theta})$ using prior $p_0(\boldsymbol{\theta})$ as follows:

$$p_\alpha(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p_0(\boldsymbol{\theta}) \prod_{t=1}^T p(\mathcal{D}_t \mid \boldsymbol{\theta})^{\alpha_t} \propto p_0(\boldsymbol{\theta})^\gamma \prod_{t=1}^T p(\boldsymbol{\theta} \mid \mathcal{D}_t)^{\alpha_t} \quad (43.31)$$

for $\gamma = 1 - \sum_{t=1}^T \alpha_t$. If we define our likelihoods and prior wisely under the generalized Bayesian framework as,

$$p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto \exp(-\ell(\boldsymbol{\theta}, \mathcal{D}_t)) \quad p_0(\boldsymbol{\theta}) \propto \exp(-\|\boldsymbol{\theta}\|_2^2), \quad (43.32)$$

we can indeed already recover the solutions to Equation 43.30 by taking the arg max of the logarithm of Equation 43.31 as follows,

$$\hat{\boldsymbol{\theta}}_{1:T} = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^p} \log p_\alpha(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^p} \sum_{t=1}^T \alpha_t \log p(\boldsymbol{\theta} \mid \mathcal{D}_t) + \gamma \log p_0(\boldsymbol{\theta}), \quad (43.33)$$

where the last equality is just a rewriting of the posterior in log space that makes the similarity to the original multitask objective explicit. This is not very practical yet, though, as we usually can not find the exact $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ and prior $p_0(\boldsymbol{\theta})$. Therefore, we will again make use of variational methods which restrict these to a specific class of distributions and thereby allow closed form expressions for the mode of the posterior.

43.5.1 Variational Model Merging

In variational methods, we restrict the posterior and prior to lie in a certain class of distributions $\mathcal{Q}$. One convenient class are exponential families, not only because we can derive practical optimizers from them, but also because we can always find the mode of a posterior in closed form to derive a merging equation. The approximate posterior based on, for example, Gaussian $q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_t, \mathcal{D}_t)$ and Gaussian prior p_0 is then:

$$q_\alpha(\boldsymbol{\theta}, \mathcal{D}_{1:T}) \propto p_0(\boldsymbol{\theta})^\gamma \prod_{t=1}^T q_t(\boldsymbol{\theta})^{\alpha_t}, \quad (43.34)$$

where q_α is again Gaussian due to conjugacy. The q_t can then be learned using a practical variational optimizer like IVON, where the posterior and prior structure determine the model merging scheme, as we again can derive the merged model $\hat{\boldsymbol{\theta}}_{1:T}$ by using

$$\hat{\boldsymbol{\theta}}_{1:T} = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^p} \log q_\alpha(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^p} \sum_{t=1}^T \alpha_t \log q_t(\boldsymbol{\theta}) + \gamma \log p_0(\boldsymbol{\theta}). \quad (43.35)$$

We will provide two examples in the following subsections and refer to Maldonado et al. [1124] for more details.

43.5.2 Recovering Averaging using Variational Merging

The posterior and prior in variational merging determine the merging equation. Here, we use simple Gaussian likelihoods with fixed variance and a standard Gaussian prior, i.e.:

$$q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_t, I) \quad p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, I). \quad (43.36)$$

Then, merging simply corresponds to weighted averaging (note the change to arg min):

$$\hat{\boldsymbol{\theta}}_{1:T} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^p} - \sum_{t=1}^T \alpha_t \log q_t(\boldsymbol{\theta}) - \gamma \log p_0(\boldsymbol{\theta}) = \sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t, \quad (43.37)$$

where the regularizer is weighted by a term $\gamma = 1 - \sum_t \alpha_t$ that vanishes due to the α_t summing to 1. In order to arrive at this result, we can use direct calculation, by noting that for the choice of likelihood and prior we can find that

$$-\log q_t(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|_2^2 + c,$$

where c is a constant under $\arg \max$, as we have chosen unit variance. The regularizer corresponds to weight-decay. Setting the derivative of Equation 43.37 to 0 leads to

$$0 = \sum_{t=1}^T \alpha_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t) + \gamma \boldsymbol{\theta} = \sum_{t=1}^T \alpha_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t) = \sum_{t=1}^T \alpha_t \boldsymbol{\theta} - \sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t \quad (43.38)$$

$$\Leftrightarrow \boldsymbol{\theta} = \sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t, \quad (43.39)$$

where we have used that $\sum_t \alpha_t = 1$ and $\gamma = 1 - \sum_t \alpha_t = 0$.

43.5.3 Uncertainty-Based Gradient Matching & Data Removal

We are often interested in merging fine-tuned LLMs $\boldsymbol{\theta}_t \in \mathbb{R}^p$, where merging generally works well if all $\boldsymbol{\theta}_t$ are initialized using the same LLM $\boldsymbol{\theta}_0 \in \mathbb{R}^p$. Then, it can make sense to account for the initialization explicitly when merging by choosing a regularizer $p_0 = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \mathbf{H}_0^{-1})$, where $\mathbf{H}_0$ is the Hessian which captures uncertainty over the parameters $\boldsymbol{\theta}_0$. Likewise, it can be beneficial to account for the uncertainty over the fine-tunings by using posteriors $q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_t, (\mathbf{H}_0 + \mathbf{H}_t)^{-1})$. This leads to the following merging equation [353]:

$$\hat{\boldsymbol{\theta}}_{1:T} = \boldsymbol{\theta}_0 + \sum_{t=1}^T \alpha_t (\mathbf{H}_0 + \sum_{t=1}^T \alpha_t \mathbf{H}_t)^{-1} (\mathbf{H}_0 + \mathbf{H}_t) (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0), \quad (43.40)$$

where using $\mathbf{H}_0 = I, \mathbf{H}_t = 0$ corresponds to Task Arithmetic [811] and using the Fisher to approximate the Hessian with $\sum_t \alpha_t = 1$ corresponds to Fisher-Weighted Averaging [1144]. The Hessians here weight each model according to their uncertainty and models with higher uncertainty, which in turn results in larger values of σ , receive less weight. This is also intuitive when remembering the IVON posterior, where larger $\mathbf{h}$ means smaller variance and stronger posterior concentration. The subtraction by $\boldsymbol{\theta}_0$ is similar to Bayes: if the q_t already contain p_0 , we would divide out the common p_0 to not overcount it, which is referred to as Bayesian Arithmetic in Khan [882]:

$$q_\alpha(\boldsymbol{\theta}, \mathcal{D}_{1:T}) = p_0(\boldsymbol{\theta}) \prod_{t=1}^T \left(\frac{q_t(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta})} \right)^{\alpha_t}. \quad (43.41)$$

While model merging often aims to combine knowledge from different models, it is also possible to remove them by using the same strategy we have used to remove the over-counted prior: we just “divide it out”. In parameter-space this corresponds to a subtraction. For example, we might be interested in removing a proportion $\mathcal{D}_t$ from an LLM, denoted by $\boldsymbol{\theta}_0$, trained on a large dataset $\mathcal{D}$ that contains $\mathcal{D}_t$. Then, we can train on $\mathcal{D}_t$ (starting from $\boldsymbol{\theta}_0$) to obtain $\boldsymbol{\theta}_t$ and perform the following merging [353]:

$$\boldsymbol{\theta}_{\setminus t} = \boldsymbol{\theta}_{\text{LLM}} - \alpha_t \mathbf{H}_{\setminus t}^{-1} (\mathbf{H}_0 + \mathbf{H}_t) (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0), \quad (43.42)$$

where $\mathbf{H}_{\setminus t}$ is calculated using $\mathcal{D} \setminus \mathcal{D}_t$ and $\boldsymbol{\theta}_0$. The method is also closely related to influence functions and, for linear regression, reduces to the influence function of Cook [327] when using $\alpha_t = 1$. We refer the interested reader to Daheim et al. [353] for further discussion and experimental results, for example, on removing toxic data from LLMs.

43.6 Concluding Remarks

In this chapter, we have reviewed various methods that can be used to apply Bayesian techniques to LLMs. First, we have shown that LLMs can be made Bayesian “post-hoc” via fine-tuning with LoRA, ensembling, and Laplace’s method. The advantage of these techniques is that existing LLMs can be reused. There is no requirement for the LLM to be trained with a Bayesian method but empirical improvements are still observed. We have then also shown that LLMs can directly be trained with Bayesian methods to learn a posterior. The information from the posterior can then be used to improve the predictions of LLMs by sampling outputs from many models sampled from the posterior and to adapt LLMs by using the uncertainty of the posterior to weight individual models.

Chapter 44

Confidence-based selective classification

Selective classification offers a principled framework to enhance the reliability of deep learning models, particularly in safety-critical settings where erroneous predictions can have serious consequences. By enabling models to abstain on uncertain inputs, selective classification explicitly controls the trade-off between *coverage* (the fraction of inputs retained rather than rejected) and *selective accuracy* (accuracy on the retained predictions). In this chapter, we systematically evaluate a broad range of uncertainty quantification (UQ) methods for confidence-based selective classification, using image recognition on CIFAR-100 as a concrete and reproducible case study. Our exploration covers novel confidence heuristics derived from softmax probabilities and prediction margins, probabilistic modeling via Bayesian Beta posterior densities, and nonparametric empirical confidence estimation, augmented with class-specific weighting schemes such as the F_1 score and geometric mean. Building on these approaches, we propose **LogitEnsemble-UQ**, a logistic-regression meta-model that integrates multiple confidence signals into a single calibrated rejection score. Evaluated at target selective accuracy levels of 80%, 85%, and 90%, this ensemble consistently achieves improved accuracy–rejection (equivalently, accuracy–coverage) trade-offs, illustrating the benefit of combining complementary uncertainty measures. Figure 44.1 provides a visual overview of the chapter: (i) how confidence estimators vary across ground-truth classes, (ii) how confidence thresholding separates correct from incorrect predictions, and (iii) the resulting selective accuracy–rejection trade-offs for the proposed ensembles.

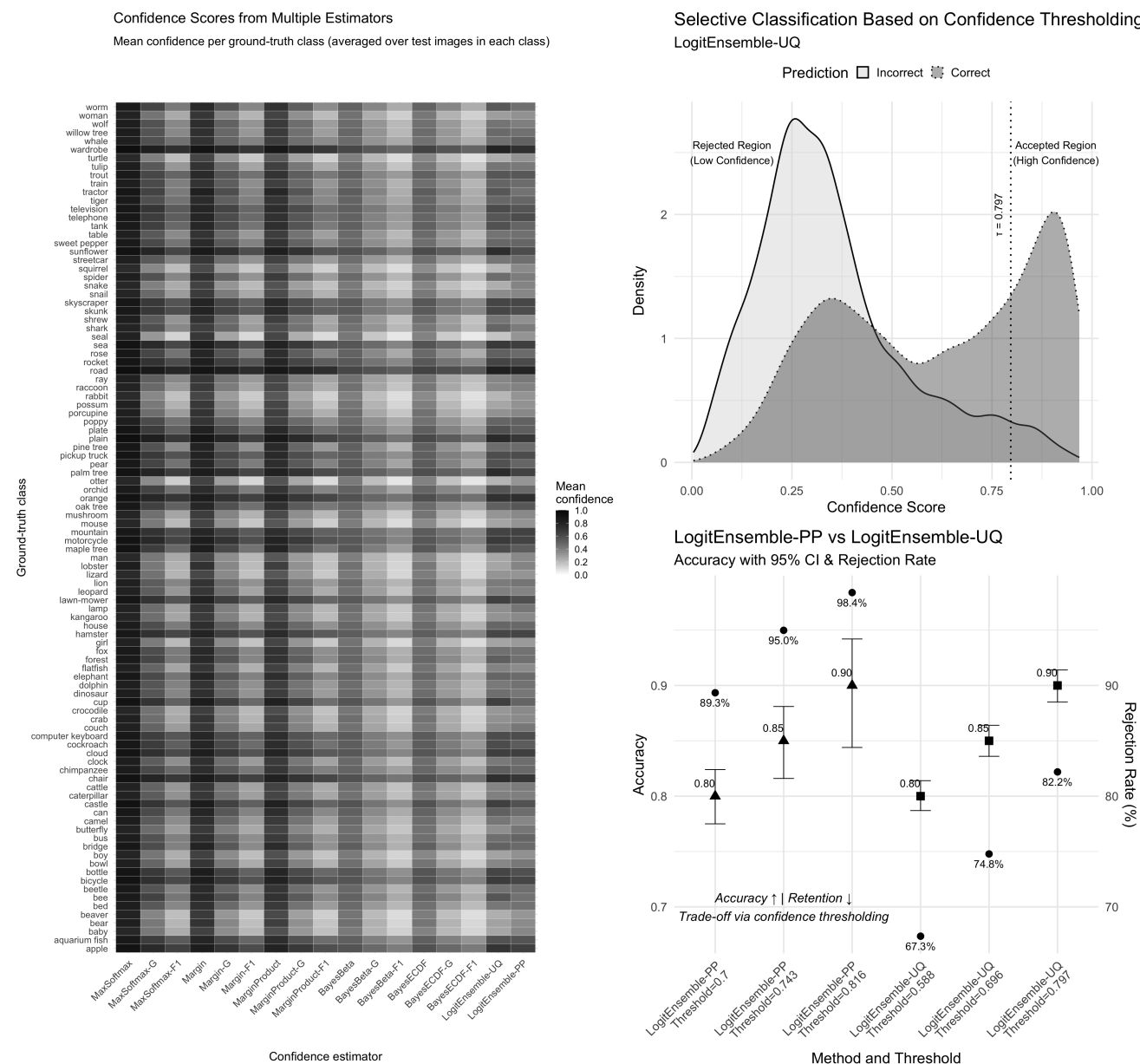


Figure 44.1: **Graphical abstract: confidence-based selective classification.** **Left:** CIFAR-100 per-class mean confidence heatmap: rows are ground-truth classes and columns are confidence estimators; each tile reports the mean confidence over all test images from that class. **Top-right:** Confidence-score densities for *LogitEnsemble-UQ*, split by correct vs. incorrect predictions. A threshold τ partitions predictions into a rejected low-confidence region and an accepted high-confidence region. **Bottom-right:** Selective classification trade-off for *LogitEnsemble-PP* and *LogitEnsemble-UQ*: selective accuracy (accuracy on accepted predictions; 95% CIs) and rejection rate at several thresholds. *LogitEnsemble-UQ* achieves higher selective accuracy at comparable (or lower) rejection, indicating that engineered uncertainty features improve confidence ranking beyond probabilities alone.

44.1 Background

Uncertainty is an inherent feature of machine learning systems, particularly in high-stakes applications where data may be noisy, incomplete, or shifted relative to the training distribution. Unlike deterministic pipelines, learned predictors must operate under ambiguity: uncertainty can arise from limited data and model misspecification (*epistemic* uncertainty) as well as irreducible noise in the observations (*aleatoric* uncertainty). Quantifying and communicating these uncertainties is central to trustworthy AI because it enables risk-aware decision-making and supports appropriate human oversight when model confidence is unreliable [566, 879].

A principled way to *act* on predictive uncertainty is to equip a classifier with a *reject option*. This idea traces back to Chow’s early work on optimal recognition with abstention, which formalized how rejecting uncertain samples can reduce error on the retained set [301, 302]. In contemporary terminology, this setting is typically referred to as *selective classification* (also called *classification with a reject option* or *classification with abstention*). The model is allowed to abstain on low-confidence inputs, thereby trading *coverage* (the proportion of inputs for which a prediction is returned) for improved reliability on the subset of *retained* predictions. Subsequent work developed theoretical and algorithmic foundations for this error–reject trade-off, including formulations based on surrogate risks and guarantees [102, 486, 1828]. Later, more general approaches considered abstention mechanisms implemented by a learned *rejector function* rather than simple thresholding of a scalar confidence score [332]. A recent survey by Hendrickx et al. synthesizes these strands, including integrated (end-to-end) methods that jointly learn predictors and rejectors [732].

A closely related—but conceptually distinct—paradigm is *learning to defer* [1119, 1201, 1224]. Whereas selective classification typically treats abstention as incurring a fixed cost (possibly sample- or class-dependent) independent of what happens after rejection, learning to defer explicitly models a downstream decision-maker (e.g., a human expert or auxiliary system) that will handle deferred inputs. In this setting, deferral is beneficial precisely when the expert is expected to outperform the model on the deferred subset. Leitão et al. discuss practical challenges and limitations of deploying deferral systems, including modeling expert behavior, distribution shift, and operational constraints in human–AI collaboration [37].

These research directions can be organized along three largely orthogonal axes:

1. **Selective classification vs. learning to defer.** Selective classification optimizes an error–reject trade-off under an assumed abstention cost [301, 486], while learning to defer optimizes the performance of a combined system (model plus expert) by exploiting complementary strengths [1119, 1201, 1224].
2. **Rejection/deferral criterion.** Many practical systems implement abstention by thresholding a scalar confidence score [301, 486]. More general approaches learn a rejector/deferral policy that can depend on richer features of the input and the model output [332, 732].
3. **Separate vs. joint (end-to-end) learning.** The base predictor and the rejection/deferral mechanism can be learned separately (two-stage systems), for example by training a classifier and then tuning a post-hoc threshold [486], or learned jointly/end-to-end (one-stage systems), where prediction and abstention are optimized together [102, 332, 732]. In learning-to-defer terminology, these correspond to two-stage and one-stage approaches [1201]. Early selective-model formulations that explicitly incorporated a learnable selection mechanism were already explored by Geifman [592], and integrated rejector approaches have since become a major theme in the literature [732].

In application-driven work, abstention and deferral have been used as practical tools to improve the reliability of machine learning outputs and to reduce human review burden by routing ambiguous cases to more reliable decision pathways. For instance, recent studies report that uncertainty-aware filtering or abstention can improve downstream operating characteristics (e.g., precision/recall-type metrics) in clinical prediction settings [274, 1086]. In sleep staging, abstention based on uncertainty measures has been used to increase agreement with reference labels, including improvements as measured by Cohen’s Kappa [867]. In medical imaging, uncertainty-driven selection has similarly been used to concentrate predictions on higher-confidence cases, improving performance metrics such as F_1 on the retained subset [614].

Large-scale deployments further motivate selective prediction as an operational mechanism. For example, selective classification for information extraction from pathology reports has been evaluated at scale with the goal of meeting stringent reliability targets while reducing manual review [1312]. Related efforts in pediatric oncology have explored uncertainty-aware automation for coding/classification from pathology text, reporting strong performance while abstaining on ambiguous cases [1821]. Across such settings, performance hinges on how well the chosen confidence score (or learned rejector/deferral policy) ranks correct predictions above incorrect ones, since threshold selection directly mediates the trade-off between coverage and reliability.

Building on these foundations, this chapter focuses on *selective classification* and systematically evaluates *post-hoc*, confidence-based uncertainty quantification methods for deep neural networks. Using CIFAR-100 as a reproducible testbed, we benchmark confidence scoring approaches—ranging from classic probability-based heuristics rooted in the reject-option literature [301, 302] to Bayesian and nonparametric calibration schemes—by their ability to satisfy user-specified target selective accuracies while maximizing retained coverage. In addition, we examine class-weighted confidence variants and propose a logistic-regression meta-model that combines multiple uncertainty signals into a single calibrated rejection score. The goal is to provide a practical, comparative perspective on

confidence scoring for selective prediction and to inform reliable deployment of uncertainty-aware deep learning systems.

44.2 Mathematical Framework

44.2.1 Problem Setup

We begin by formalizing the supervised classification setting in which selective classification and uncertainty quantification (UQ) will be applied.

Let

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

denote a dataset of N labeled samples, where $x_i \in \mathcal{X}$ is an input from the domain $\mathcal{X}$ (e.g., images), and $y_i \in \mathcal{Y} = \{1, \dots, C\}$ is a categorical label from a set of C mutually exclusive classes. In our CIFAR-100 case study, $C = 100$. We assume access to a trained base classifier

$$f_\theta : \mathcal{X} \rightarrow \Delta^{C-1},$$

parameterized by $\theta \in \Theta$, which maps each input to a probability vector lying on the $(C-1)$ -dimensional probability simplex

$$\Delta^{C-1} = \left\{ \mathbf{p} \in \mathbb{R}^C \mid p_c \geq 0 \ \forall c, \quad \sum_{c=1}^C p_c = 1 \right\}.$$

The simplex constraint guarantees that outputs are interpretable as a valid probability distribution.

For each input x_i , the classifier outputs

$$\mathbf{p}_i = f_\theta(x_i) = [p_{i,1}, p_{i,2}, \dots, p_{i,C}]^\top,$$

where $p_{i,c}$ denotes the model's estimated probability that x_i belongs to class c . The predicted class label is obtained via the maximum a posteriori (MAP) decision rule:

$$\hat{y}_i = \arg \max_{c \in \mathcal{Y}} p_{i,c}.$$

Intuition: The classifier produces a probability distribution over classes, but these numbers are not necessarily well-calibrated; a 90% prediction may not actually be correct 90% of the time. Selective classification seeks to leverage this distribution to decide when to trust the model.

44.2.2 Confidence Scoring for Selective Classification

In conventional classification, every prediction is returned without qualification. Selective classification changes this: for each prediction, we compute a *confidence score* and compare it to a threshold to decide whether to retain the prediction or not.

Formally, a scoring function is a mapping

$$s : \mathcal{X} \rightarrow \mathbb{R}, \quad s_i = s(x_i),$$

where higher s_i indicates greater confidence in $\hat{y}_i$ being correct i.e., matches the true label y_i . The ranking induced by s determines the trade-off between coverage (fraction of samples retained) and risk (error rate on retained samples).

We now outline several confidence scoring strategies, each grounded in either direct probability analysis or a probabilistic generative model.

MaxSoftmax

Perhaps the most widely used confidence score, the *MaxSoftmax* is simply the highest predicted class probability:

$$s_i = \max_c p_{i,c}.$$

This value corresponds directly to the model’s own stated belief that its most likely class is correct. If the model produces a well-calibrated probability distribution, s_i can be interpreted as an estimate of the probability of correctness.

Intuition: A high MaxSoftmax value indicates that the model strongly favors one class over all others. For example, $s_i = 0.99$ means the model assigns 99% probability to a single class and only 1% to all others combined. This is a simple, fast-to-compute measure that often works surprisingly well as a first-pass uncertainty estimate. However, it can be overconfident in overparameterized neural networks, especially when probabilities are not perfectly calibrated.

Margin

The *Margin* score focuses on the gap between the top two predicted probabilities:

$$s_i = p_{i,(1)} - p_{i,(2)},$$

where $p_{i,(1)} \geq p_{i,(2)} \geq \dots \geq p_{i,(C)}$ are the predicted probabilities sorted from largest to smallest.

Intuition: Rather than only considering the top class, the margin measures how decisively it beats the runner-up. A large margin implies a clear “winner” among classes, while a small margin means the decision is contested between at least two plausible labels. This is particularly useful in cases where MaxSoftmax is high but two classes are nearly tied — a sign that the model may still be uncertain despite a confident top probability.

MarginProduct

The *MarginProduct* combines the strengths of both the MaxSoftmax and Margin scores:

$$s_i = (p_{i,(1)} - p_{i,(2)}) \cdot p_{i,(1)}.$$

Here, the margin captures the decisiveness of the prediction, while $p_{i,(1)}$ captures its absolute confidence level.

Intuition: This hybrid score rewards predictions that are both *highly confident* (large $p_{i,(1)}$) and *decisive* (large margin). If either the top probability $p_{i,(1)}$ is low (low confidence) or the margin $p_{i,(1)} - p_{i,(2)}$ is small (runner-up close), the score decreases accordingly. This makes MarginProduct more conservative than either MaxSoftmax or Margin alone, often leading to better separation between correct and incorrect predictions in practice.

BayesBeta

We take a generative perspective, modeling the top predicted probability $p_{i,(1)}$ under two separate Beta distributions:

$$p_{i,(1)} \sim \begin{cases} \text{Beta}(\alpha_{\text{corr}}, \beta_{\text{corr}}), & \hat{y}_i = y_i, \\ \text{Beta}(\alpha_{\text{incorr}}, \beta_{\text{incorr}}), & \hat{y}_i \neq y_i. \end{cases}$$

Let f_{corr} and f_{incorr} denote the corresponding probability density functions, and let t_{prior} denote the prior probability that a prediction is correct. By Bayes’ theorem, the posterior probability of correctness given $p_{i,(1)}$ is:

$$s_i = \frac{t_{\text{prior}} \cdot f_{\text{corr}}(p_{i,(1)})}{t_{\text{prior}} \cdot f_{\text{corr}}(p_{i,(1)}) + (1 - t_{\text{prior}}) \cdot f_{\text{incorr}}(p_{i,(1)})}.$$

Intuition: The BayesBeta method explicitly asks: “Given that my model assigned a highest class probability of $p_{i,(1)}$, how likely is it that this prediction is actually correct?” Instead of relying on $p_{i,(1)}$ in isolation as a raw confidence proxy, we model how this quantity behaves *differently* for correct versus incorrect predictions. In many models, correct predictions tend to have $p_{i,(1)}$ values concentrated near 1, while incorrect predictions show a broader spread toward lower values. The Beta distribution is well-suited here because: (1) It naturally models probabilities in the $[0, 1]$ interval. (2) Its shape is highly flexible — it can represent sharp peaks, flat distributions, or skewed shapes. By fitting one Beta distribution to the correct cases and another to the incorrect cases, we effectively learn two distinct “signatures” of top-probability behavior. Bayes’ theorem then lets us combine these likelihoods with a prior belief about correctness (t_{prior}) to produce a calibrated, probabilistic estimate of correctness. The result is a confidence score that is *interpretable*, *probabilistically grounded*, and capable of capturing subtle distributional differences that simple thresholds on $p_{i,(1)}$ might miss.

BayesECDF

The BayesECDF method follows the same Bayesian reasoning as BayesBeta, but removes the parametric assumption on the shape of the distributions. Instead of fitting Beta distributions, we directly estimate the empirical cumulative distribution functions (ECDFs) of the top predicted probability $p_{i,(1)}$ for correct and incorrect predictions:

$$F_{\text{corr}}(p) = \frac{1}{N_{\text{corr}}} \sum_{i=1}^N \mathbb{I}[\hat{y}_i = y_i \wedge p_{i,(1)} \leq p],$$

$$F_{\text{incorr}}(p) = \frac{1}{N_{\text{incorr}}} \sum_{i=1}^N \mathbb{I}[\hat{y}_i \neq y_i \wedge p_{i,(1)} \leq p],$$

where N_{corr} and N_{incorr} are the total counts of correct and incorrect predictions, respectively.

From these ECDFs, we approximate the probability density functions $f_{\text{corr}}(p)$ and $f_{\text{incorr}}(p)$ by taking finite differences:

$$f_{\text{corr}}(p) \approx \frac{d}{dp} F_{\text{corr}}(p), \quad f_{\text{incorr}}(p) \approx \frac{d}{dp} F_{\text{incorr}}(p).$$

These density estimates are then plugged into Bayes' theorem exactly as in BayesBeta to yield the posterior probability of correctness:

$$s_i = \frac{t_{\text{prior}} \cdot f_{\text{corr}}(p_{i,(1)})}{t_{\text{prior}} \cdot f_{\text{corr}}(p_{i,(1)}) + (1 - t_{\text{prior}}) \cdot f_{\text{incorr}}(p_{i,(1)})}.$$

Intuition: While BayesBeta assumes that $p_{i,(1)}$ follows a Beta distribution for both correct and incorrect predictions, BayesECDF makes no such assumption. This is valuable when the empirical shapes of these distributions are irregular — for example, multimodal patterns or sharp spikes that the Beta family cannot easily capture. The ECDF-based approach lets the data “speak for itself.” By counting how often the top predicted probability falls below a given value p , separately for correct and incorrect cases, we obtain a complete nonparametric description of its distribution. From there, the same Bayesian logic applies: we compare the likelihood of observing $p_{i,(1)}$ under each case (correct vs. incorrect) and combine it with the prior probability of correctness. This flexibility comes with a trade-off: while BayesECDF can more faithfully represent unusual empirical patterns, it may require more data to produce smooth, stable estimates of the densities. Nevertheless, in large-scale classification problems, its ability to model complex score distributions without constraining them to a fixed parametric form can lead to more accurate and better-calibrated confidence estimates.

Class-Weighted Variants

The confidence scores described so far treat all classes as equally difficult. However, in many real-world classification tasks — especially those with class imbalance or heterogeneous complexity — some classes are consistently easier for the model, while others are systematically more challenging. To account for these differences, we introduce *class-weighted variants* of any base confidence score.

Formally, given a base confidence score s_i and a per-class weight $w_{\hat{y}_i}$ corresponding to the predicted class $\hat{y}_i$, we define the reweighted score:

$$s_i^{(w)} = s_i \cdot w_{\hat{y}_i}.$$

This weighting scheme increases the effective confidence for classes that the model typically predicts accurately, and decreases it for classes where the model is less reliable. In practice, class-weighted variants are ranking scores and thresholds become method-specific.

Two weighting metrics are considered:

- **Geometric Mean (G-mean):** This metric balances the true negative rate and true positive rate for class k , providing a balanced measure of performance across both correctly rejecting negatives and correctly identifying positives:

$$G_k = \sqrt{\frac{\text{TN}_k}{\text{TN}_k + \text{FP}_k} \cdot \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}}.$$

- **F1-score:** The harmonic mean of precision and recall for class k , capturing the trade-off between avoiding false positives and avoiding false negatives:

$$F1_k = \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k},$$

where

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}, \quad \text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}.$$

Here, k indexes the class, and TP_k , FP_k , TN_k , FN_k are counts from the per-class confusion matrix.

Intuition: Consider a classifier trained on a dataset where some classes are rare or visually ambiguous. If the model predicts such a difficult class with high probability, a raw confidence score might still overestimate the likelihood of correctness. By scaling scores according to the model’s estimated per-class performance on a held-out validation (calibration) set, we make the confidence estimates more *class-aware*.

For example, if class A has a high F1-score, we trust the model’s predictions for A more and boost its scores; if class B has low G-mean, we discount its scores accordingly. This adjustment is especially valuable in imbalanced datasets, as it mitigates the risk of overconfidence on poorly represented classes while allowing the model to be bolder where it has consistently strong performance.

LogitEnsemble-UQ

Rather than relying on a single uncertainty score, we can combine multiple complementary scores into a unified confidence estimate. To achieve this, we construct a feature vector $\mathbf{z}_i$ for each sample x_i containing all base confidence scores and their class-weighted variants:

$$\mathbf{z}_i = \left[\text{MaxSoftmax}_i, \dots, \text{BayesBeta}_i, \text{BayesECDF}_i, \text{MaxSoftmax}_i^{(G)}, \dots, \text{BayesECDF}_i^{(F1)} \right]^\top.$$

A logistic regression model is then trained on these vectors to produce a single scalar confidence score:

$$s_i = \sigma(\mathbf{w}^\top \mathbf{z}_i + b),$$

where $\sigma(t) = \frac{1}{1+e^{-t}}$ is the logistic sigmoid function, and $(\mathbf{w}, b)$ are learned parameters.

Intuition: Each base score (e.g., MaxSoftmax, Margin, BayesBeta, etc.) captures a different aspect of model uncertainty. For example, MaxSoftmax focuses purely on the top predicted probability, while Margin considers the gap between top predictions, and Bayes-based methods incorporate distributional calibration. Individually, each score is imperfect—some are better for certain classes or accuracy ranges, others for different operating points. By feeding all these scores into a logistic regression meta-model, the system learns an optimal way to *weight* and *combine* them. Logistic regression is well-suited here because: (1) It outputs a value between 0 and 1, naturally interpretable as a probability of correctness. (2) It can learn linear combinations that emphasize informative features and downweight noisy ones. (3) It is simple and transparent, making the learned coefficients interpretable—allowing us to see which base scores contribute most. The result is an *ensemble confidence score* that generally outperforms any individual score in ranking predictions from most to least reliable. In practice, this approach often yields lower rejection rates for the same target accuracy, as the meta-model learns to exploit synergies between different uncertainty measures.

LogitEnsemble-PP

As a simpler baseline, we can bypass engineered uncertainty features altogether and feed the raw class probability vector $\mathbf{p}_i$ directly into a logistic regression model:

$$s_i = \sigma(\mathbf{w}^\top \mathbf{p}_i + b),$$

where $\sigma(\cdot)$ is the logistic sigmoid function and $(\mathbf{w}, b)$ are learned parameters. L_2 -regularized logistic regression can be applied to reduce overfitting to spurious patterns in the class probability vector $\mathbf{p}_i$, such as calibration biases and dataset-specific probability correlations.

Intuition: The vector $\mathbf{p}_i$ contains the model’s full posterior distribution over classes. While this raw representation implicitly encodes uncertainty (e.g., a sharp peak for confident predictions, a flatter distribution for uncertain ones), it does not explicitly combine or transform this information in the way the engineered features do. The logistic regression here learns to map these probability patterns directly to an estimated probability of correctness. This baseline is valuable for two reasons: (1) It isolates the benefit of using *only* the base model’s probability outputs, without additional derived features. (2) It provides a clean benchmark to evaluate whether the engineered uncertainty features (and their combinations, as in LogitEnsemble-UQ) meaningfully improve selective classification performance.

Because $\mathbf{p}_i$ is high-dimensional (one feature per class), this method may capture nuanced patterns in the output distribution that simple scalar scores cannot. However, without targeted uncertainty descriptors (e.g., MaxSoftmax, Margin, BayesBeta, etc.), it may fail to fully leverage the structure of model uncertainty—making it a possibly informative but less robust alternative to LogitEnsemble-UQ.

44.2.3 Selective Classification by Thresholding

Once a confidence score s_i has been assigned to each prediction, selective classification proceeds by retaining only those predictions deemed sufficiently reliable. This is achieved via a *thresholding* operation: we specify a confidence threshold $\tau \in \mathbb{R}$ and reject predictions whose scores fall below it.

Formally, let $s(x) \in \mathbb{R}$ denote a confidence score assigned by the model to an input feature vector x . Selective classification is obtained by *thresholding* this score: given a threshold τ , we retain predictions whose confidence is at least τ , and reject the rest.

Formally, the binary *selection function* is

$$g_\tau(x) = \begin{cases} 1, & s(x) \geq \tau, \\ 0, & \text{otherwise,} \end{cases}$$

where $g_\tau(x) = 1$ indicates that the prediction for x is *retained* and $g_\tau(x) = 0$ indicates that it is *rejected*.

Intuition: By adjusting τ , we control the *coverage–accuracy trade-off*. A low threshold retains most predictions (high coverage) but risks keeping low-confidence cases, potentially lowering accuracy. Conversely, a high threshold retains only the most confident predictions (low coverage), often boosting accuracy at the cost of discarding more samples.

44.2.4 Coverage and Rejection Rate

Coverage quantifies the proportion of predictions that are *retained* under a given confidence threshold τ :

$$\text{coverage}(\tau) = \frac{1}{N} \sum_{i=1}^N g_\tau(x_i),$$

where $g_\tau(x_i) \in \{0, 1\}$ is the selection function.

The *rejection rate* is simply the fraction of predictions that are discarded:

$$\text{rejection rate}(\tau) = 1 - \text{coverage}(\tau).$$

Intuition: Coverage and rejection rate are two sides of the same coin. A high coverage means that most predictions are kept, which maximizes throughput but may include low-confidence cases. A high rejection rate means that only the most confident predictions survive — improving reliability but potentially reducing the usable output size.

44.2.5 Selective Accuracy and Risk

Selective accuracy measures the fraction of correct predictions *among retained samples*:

$$\text{selective accuracy}(\tau) = \frac{\sum_{i=1}^N \mathbb{I}[\hat{y}_i = y_i] \cdot g_\tau(x_i)}{\sum_{i=1}^N g_\tau(x_i)}.$$

The corresponding *selective risk* is

$$\text{selective risk}(\tau) = 1 - \text{selective accuracy}(\tau).$$

In this work, selective accuracy is reported with 95% binomial proportion confidence intervals using the Clopper–Pearson method [314].

Intuition: Selective accuracy answers the question: “Given that I only keep high-confidence predictions, how often am I right?” Selective risk flips the perspective: “Among the predictions I keep, how often am I wrong?” These two metrics capture the reliability of the retained set, independent of how many samples are discarded.

44.2.6 Target Accuracy Thresholding

For a desired accuracy target $\alpha \in (0, 1)$, we choose the smallest threshold τ_α satisfying:

$$\tau_\alpha = \min\{\tau : \text{selective accuracy}(\tau) \geq \alpha\}.$$

This approach enforces a pre-specified accuracy guarantee, even if it requires discarding a large portion of the data. In practice, we sweep thresholds over unique score values (or quantiles) on the validation set and pick the smallest τ meeting the constraint.

Intuition: Target accuracy thresholding flips the usual problem: instead of asking “How accurate am I at a given coverage?”, it asks “How much data can I keep while meeting my accuracy goal?” The resulting τ_α adapts the system’s selectiveness to the desired reliability standard.

44.2.7 Risk–Coverage Curves

By varying τ from low to high, we trace a *risk–coverage curve*, plotting coverage on the x -axis and selective risk on the y -axis:

$$\text{coverage} = 1 - \text{rejection rate}.$$

An ideal method shows a steep drop in risk as coverage decreases, indicating that it quickly eliminates uncertain predictions while retaining as many correct ones as possible.

Intuition: Risk–coverage curves provide a global view of how well a confidence score separates good predictions from bad ones. A shallow curve means the method struggles to find high-confidence subsets without also discarding many correct predictions. A steep curve means that even a small reduction in coverage yields a large drop in risk — a hallmark of an effective uncertainty estimator.

44.3 Case Example on CIFAR-100

To concretely illustrate the selective classification framework introduced in Section 44.2.2, we evaluate a modern image classifier on the CIFAR-100 benchmark. The CIFAR-100 dataset consists of $N = 50,000$ labeled images, each belonging to one of $C = 100$ classes. We adopt a split of 40,000 images for training the ResNet and 10,000 images for post-hoc calibration and evaluation. All selective-classification components that require labeled data and the operating thresholds τ_α are estimated on this 10,000-image split, and results are reported on the same split. Consequently, the reported numbers should be interpreted as calibration-set performance rather than performance on an independent test set.

The base model f_θ is a ResNet [721, 1789], trained to output a vector of class probabilities for each input x_i :

$$\mathbf{p}_i = f_\theta(x_i) = [p_{i,1}, p_{i,2}, \dots, p_{i,C}]^\top,$$

where $p_{i,c}$ represents the model’s estimated probability that x_i belongs to class c , with $\sum_{c=1}^C p_{i,c} = 1$. Predicted labels are assigned by maximum a posteriori (MAP) decision:

$$\hat{y}_i = \arg \max_{c \in \mathcal{Y}} p_{i,c}.$$

A natural baseline confidence score for selective classification is simply the highest predicted probability:

$$s_i^{\text{base}} = \max_c p_{i,c} = p_{i,(1)},$$

where $p_{i,(1)}$ denotes the largest component of $\mathbf{p}_i$.

Following the general framework, we compute a range of confidence scores $s_i = s(x_i)$ — including MaxSoftmax, Margin, MarginProduct, BayesBeta, BayesECDF, and their class-weighted variants — to quantify uncertainty and rank predictions by reliability. For each method, we identify the smallest threshold τ_α that achieves a target *selective accuracy* $\alpha \in \{0.80, 0.85, 0.90\}$:

$$\tau_\alpha = \min\{\tau \in \mathbb{R} : \text{selective accuracy}(\tau) \geq \alpha\}.$$

Here, *selective accuracy* is the accuracy computed over only the retained predictions, i.e., those with $s_i \geq \tau_\alpha$.

Quantitative Performance at Target Accuracies

Tables 44.1–44.3 report the results for all methods at each target accuracy. Each row shows: (1) the achieved selective accuracy with a 95% confidence interval, (2) the rejection rate (percentage of samples discarded), (3) the total set size used to compute the metrics (*support*), and (4) the learned or selected threshold τ_α .

The baseline model (no rejection) achieves $\approx 47.6\%$ accuracy on CIFAR-100, underscoring the dataset’s difficulty. To meet higher accuracy targets, a substantial proportion of low-confidence predictions must be rejected. At the most stringent target (90%), most scoring methods reject roughly 82%–83% of samples, retaining only the most certain predictions.

A key observation is that while many methods meet the same target accuracy by construction (via threshold selection), their *rejection rates* vary considerably, reflecting differences in how well each score ranks correct predictions above incorrect ones. Across the *unweighted* scalar heuristics, MaxSoftmax, Margin, and MarginProduct exhibit very similar behavior on CIFAR-100, suggesting that for this model the top probability $p_{(1)}$ already captures most of the separability information available in the softmax vector, and that incorporating the runner-up probability via the margin provides only marginal additional benefit at these operating points. The Bayesian calibration methods (BayesBeta and BayesECDF) also perform comparably to these simple heuristics, indicating that modeling $p_{(1)}$ via a parametric Beta family or a nonparametric ECDF does not substantially change the induced ranking on this test bed, even though these approaches are appealing because they output a more interpretable, “probability-of-correctness” style confidence score.

Across all three target accuracies, the class-weighted variants (G-mean and F_1) are generally *less effective* than their unweighted counterparts, typically requiring higher rejection to meet the same accuracy constraint. One plausible explanation is that CIFAR-100 is approximately class-balanced and relatively small, so class-level weights derived from per-class performance do not primarily correct imbalance; instead, they act as a multiplicative rescaling of the confidence score, $s_i^{(w)} = s_i \cdot w_{y_i}$, which can distort the global ranking that selective classification depends on. In particular, downweighting predictions from empirically “hard” classes can cause many otherwise-correct predictions to fall below the threshold τ_α , increasing rejection without a compensating improvement in separating correct from incorrect cases. In addition, with $C = 100$ classes and limited samples per class, per-class metrics such as F_1 and G-mean can be noisy, and this estimation noise can propagate directly into the weighted confidence score.

The logistic regression ensemble with engineered uncertainty features (LogitEnsemble-UQ) consistently yields the *lowest* rejection rate at all targets, indicating a more favorable accuracy–coverage trade-off and a stronger ability to combine complementary cues (absolute confidence, decisiveness, and calibrated posteriors) into a single ranking signal. In contrast, the probability-based logistic regression baseline (LogitEnsemble-PP) is far less efficient, often rejecting $\geq 95\%$ of the data to satisfy the same target accuracy. A plausible reason is that using the full C -dimensional probability vector as features can encourage the meta-model to fit idiosyncratic, class-specific probability patterns that do not translate into a clean global “correct vs. incorrect” separation, whereas the engineered features in LogitEnsemble-UQ act as lower-dimensional, uncertainty-focused summaries that are more directly aligned with the selective classification objective.

We emphasize that CIFAR-100 is used here primarily as a controlled and reproducible *test bed* for comparing uncertainty scores under a common protocol; it is not meant to be the setting where all variants necessarily shine. In larger and more heterogeneous real-world datasets—especially those with pronounced class imbalance, class-dependent difficulty, label noise, or differing misclassification costs—class-aware weighting and calibration can become more valuable. In such regimes, class-weighted confidence may better reflect the fact that “a score of 0.9” can correspond to different empirical reliability depending on the predicted class, potentially enabling higher coverage at a fixed reliability target. Therefore, the weaker performance of the weighted variants on CIFAR-100 should be interpreted as evidence that simple global class-weighting is not universally beneficial, rather than as a general negative result about class-aware selective classification.

Trade-off Between Coverage and Accuracy

Figure 44.2 presents the achieved accuracy (with 95% confidence intervals) as a function of the rejection rate for all uncertainty scoring methods, stratified by the three target accuracy levels (80%, 85%, and 90%). Each panel corresponds to one target accuracy, and within each panel, points represent different methods, grouped by scoring family (linetype) and type (shape). Horizontal position reflects the fraction of predictions rejected, while vertical position reflects the actual accuracy on the retained set. Across all targets, higher rejection rates generally correspond to higher accuracies — confirming the expected trade-off where discarding low-confidence predictions improves reliability. However, the *efficiency* of this trade-off differs across methods. Methods near the

left edge (low rejection) achieve the most favorable balance, meeting the target accuracy with minimal discarded data. Here, the **LogitEnsemble-UQ** stands out: in all three panels, it reaches the target with substantially fewer rejections compared to both simple probability-based scores (e.g., **MaxSoftmax**) and more complex Bayesian or class-weighted variants. In contrast, the **LogitEnsemble-PP** baseline often lies far to the right, indicating it rejects the majority of predictions to meet the same target — a sign of weaker discrimination between correct and incorrect cases.

Risk–Coverage Perspective

Figure 44.3 compares the **LogitEnsemble-UQ** and **LogitEnsemble-PP** methods through their *risk–coverage* curves. Here, *coverage* is the fraction of predictions retained after thresholding by confidence, while *selective risk* is the error rate computed only on those retained predictions. The primary *y*-axis (left) displays selective risk, and the secondary *y*-axis (right) shows the equivalent selective accuracy for reference. An ideal curve hugs the lower right corner — maintaining low risk (high accuracy) while preserving high coverage. Across all thresholds, **LogitEnsemble-UQ** exhibits a steeper decline in risk as coverage decreases, meaning it is more effective at discarding low-confidence errors while keeping correct predictions. By contrast, **LogitEnsemble-PP** follows a more gradual slope, requiring a greater loss of coverage to achieve comparable risk levels. This difference reflects the value of incorporating engineered uncertainty features into the meta-model: **LogitEnsemble-UQ** can separate correct from incorrect predictions more sharply, enabling tighter control over the accuracy–coverage trade-off.

Density Separation of Correct vs. Incorrect Predictions

Figure 44.4 examines how confidence scores are distributed for correct and incorrect predictions under **LogitEnsemble-UQ** and **LogitEnsemble-PP**, at both baseline and target accuracy thresholds (80%, 85%, and 90%). Each panel shows a kernel density estimate of confidence scores, with correct predictions shaded separately from incorrect ones. A desirable pattern is a clear separation — correct predictions clustered toward high confidence, and incorrect predictions concentrated at low confidence. For all accuracy targets, **LogitEnsemble-UQ** exhibits a sharper separation: the correct-prediction density peaks closer to $s_i = 1$, while the incorrect-prediction density is shifted markedly toward lower scores. This separation remains consistent at higher target accuracies, reflecting the model’s ability to identify and reject uncertain predictions. In contrast, **LogitEnsemble-PP** shows greater overlap between the two densities, which limits its ability to achieve low error rates without substantial rejection. These density plots visually confirm the calibration advantage of **LogitEnsemble-UQ** observed in the previous risk–coverage analysis.

Practical Implication

These results demonstrate that selective classification is not merely a theoretical construct — it is a practical and versatile mechanism for managing the trade-off between coverage and reliability in real-world systems. In many operational contexts, the ability to abstain from making a prediction can be as important as making one. By selectively rejecting uncertain cases, a model can achieve substantially higher accuracy on the subset of predictions it retains, thereby offering a built-in safeguard against costly errors. The value of this capability depends on the application domain. In safety-critical environments such as medical diagnosis, autonomous driving, or industrial quality control, a single incorrect prediction may have severe consequences, making a high-confidence subset preferable even at the cost of reduced coverage. Conversely, in high-throughput settings where human review is possible but expensive — for example, document triage in legal discovery or content moderation at scale — lowering the rejection rate while still meeting strict accuracy thresholds can translate directly into lower operational costs and faster turnaround times. Our results show that by incorporating richer and more diverse uncertainty features, such as those used in **LogitEnsemble-UQ**, it is possible to meet stringent target accuracies while discarding fewer samples compared to simpler confidence scores. This efficiency means that the model’s abstention mechanism is more selective, rejecting only those cases most likely to be incorrect, rather than broadly discarding borderline predictions. Such precision in abstention decisions not only improves downstream efficiency but also enhances user trust by making the system’s predictions more predictable and reliable.

44.4 Conclusion

This chapter has presented a comprehensive treatment of selective classification through the lens of uncertainty-aware rejection. By equipping models with the ability to withhold low-confidence predictions, we move toward safer, more controllable AI systems—particularly crucial when mistakes carry high costs. In safety-critical domains such

Table 44.1: Selective classification performance at 90% target accuracy.

Method	Accuracy [95% CI]	Rejection Rate (%)	Support	Threshold
Baseline	0.476 [0.466, 0.485]	0.00	10,000	—
MaxSoftmax	0.901 [0.886, 0.915]	83.06	10,000	0.999850
Margin	0.901 [0.885, 0.914]	83.20	10,000	0.999740
MarginProduct	0.901 [0.885, 0.914]	83.10	10,000	0.999580
BayesBeta	0.900 [0.885, 0.914]	82.97	10,000	0.710200
BayesECDF	0.900 [0.885, 0.914]	83.07	10,000	0.755619
MaxSoftmax-G	0.900 [0.870, 0.925]	95.08	10,000	0.853230
Margin-G	0.901 [0.875, 0.923]	93.55	10,000	0.836330
MarginProduct-G	0.901 [0.875, 0.923]	93.44	10,000	0.820560
BayesBeta-G	0.900 [0.885, 0.914]	83.45	10,000	0.508990
BayesECDF-G	0.900 [0.883, 0.916]	86.24	10,000	0.572145
MaxSoftmax-F1	0.900 [0.862, 0.931]	96.89	10,000	0.749990
Margin-F1	0.900 [0.868, 0.927]	95.78	10,000	0.729300
MarginProduct-F1	0.900 [0.870, 0.925]	95.10	10,000	0.709650
BayesBeta-F1	0.900 [0.884, 0.916]	86.04	10,000	0.406490
BayesECDF-F1	0.900 [0.882, 0.917]	88.25	10,000	0.460190
LogitEnsemble-UQ	0.900 [0.885, 0.914]	82.19	10,000	0.796690
LogitEnsemble-PP	0.900 [0.844, 0.942]	98.39	10,000	0.816200

Table 44.2: Selective classification performance at 85% target accuracy.

Method	Accuracy [95% CI]	Rejection Rate (%)	Support	Threshold
Baseline	0.476 [0.466, 0.485]	0.00	10,000	—
MaxSoftmax	0.850 [0.835, 0.865]	77.15	10,000	0.999150
Margin	0.850 [0.835, 0.865]	77.56	10,000	0.998630
MarginProduct	0.850 [0.835, 0.865]	77.36	10,000	0.997760
BayesBeta	0.851 [0.835, 0.865]	77.12	10,000	0.684890
BayesECDF	0.851 [0.835, 0.865]	77.35	10,000	0.751891
MaxSoftmax-G	0.850 [0.829, 0.869]	87.00	10,000	0.775310
Margin-G	0.851 [0.833, 0.867]	83.14	10,000	0.754990
MarginProduct-G	0.850 [0.833, 0.866]	82.00	10,000	0.737350
BayesBeta-G	0.850 [0.836, 0.864]	75.73	10,000	0.462390
BayesECDF-G	0.850 [0.835, 0.865]	77.24	10,000	0.516190
MaxSoftmax-F1	0.850 [0.828, 0.870]	88.45	10,000	0.647530
Margin-F1	0.850 [0.830, 0.869]	86.57	10,000	0.621790
MarginProduct-F1	0.850 [0.832, 0.868]	84.30	10,000	0.597070
BayesBeta-F1	0.850 [0.834, 0.866]	79.34	10,000	0.349690
BayesECDF-F1	0.850 [0.835, 0.865]	77.79	10,000	0.360490
LogitEnsemble-UQ	0.850 [0.836, 0.864]	74.78	10,000	0.696290
LogitEnsemble-PP	0.850 [0.816, 0.881]	94.98	10,000	0.742600

as medical diagnosis, autonomous driving, or fraud detection, selective classification improves system reliability by ensuring that only high-confidence outputs are automated, while uncertain cases can be deferred to human experts or auxiliary systems. This approach mirrors rational decision-making under uncertainty and provides a principled framework to balance accuracy, coverage, and trustworthiness.

We introduced a unified mathematical framework for confidence scoring and selection, spanning classical and probabilistic scoring functions, and examined their performance under various rejection-aware strategies, including class-specific weighting schemes. Our empirical analysis revealed how different confidence estimators influence the trade-off between accuracy and coverage, and demonstrated that ensemble-based scoring can deliver better calibration and robustness. Crucially, confidence-based thresholding allows models to meet user-specified accuracy targets while minimizing the proportion of filtered samples—an asset in risk-sensitive deployments where precision is non-negotiable.

Yet, selective classification comes with inherent trade-offs. Reduced coverage may be unacceptable in high-throughput applications, and the approach depends critically on the quality of uncertainty estimates. Poor calibration can cause the system to filter easy cases or misclassify difficult ones with unwarranted confidence. Moreover, real-world deployments must contend with operational costs of filtering—delays, human intervention, and potential inequities if filtering rates differ across subpopulations.

Looking ahead, a promising research direction is to move from post-hoc filtering to end-to-end training of abstention-aware models, where classification and selection are jointly optimized. In this context, the confidence scoring functions developed in this chapter have broader utility beyond selective classification; they can be leveraged in active learning paradigms to identify the most informative samples for annotation, thereby improving data efficiency and accelerating model improvement. This synergy highlights the wider impact of principled uncertainty quantification in building adaptive, data-efficient, and trustworthy AI systems.

Selective classification is not merely a performance enhancement; it is a foundational framework for building adaptive, accountable, and risk-aware AI. As the field advances, integrating uncertainty estimation and decision-

Table 44.3: Selective classification performance at 80% target accuracy.

Method	Accuracy [95% CI]	Rejection Rate (%)	Support	Threshold
Baseline	0.476 [0.466, 0.485]	0.00	10,000	—
MaxSoftmax	0.800 [0.785, 0.814]	70.07	10,000	0.996569
Margin	0.800 [0.785, 0.814]	70.59	10,000	0.994400
MarginProduct	0.800 [0.785, 0.814]	70.10	10,000	0.990470
BayesBeta	0.800 [0.785, 0.814]	70.07	10,000	0.635290
BayesECDF	0.800 [0.785, 0.814]	70.13	10,000	0.741190
MaxSoftmax-G	0.800 [0.784, 0.816]	75.42	10,000	0.721790
Margin-G	0.800 [0.784, 0.816]	74.12	10,000	0.702200
MarginProduct-G	0.800 [0.785, 0.815]	72.03	10,000	0.682580
BayesBeta-G	0.800 [0.786, 0.814]	67.85	10,000	0.413990
BayesECDF-G	0.800 [0.786, 0.814]	68.40	10,000	0.441990
MaxSoftmax-F1	0.800 [0.782, 0.818]	80.12	10,000	0.577079
Margin-F1	0.800 [0.783, 0.817]	78.30	10,000	0.541160
MarginProduct-F1	0.800 [0.783, 0.817]	77.22	10,000	0.519430
BayesBeta-F1	0.801 [0.786, 0.815]	70.21	10,000	0.300890
BayesECDF-F1	0.800 [0.785, 0.814]	69.45	10,000	0.304390
LogitEnsemble-UQ	0.801 [0.787, 0.814]	67.35	10,000	0.587890
LogitEnsemble-PP	0.800 [0.775, 0.824]	89.34	10,000	0.700090

theoretic reasoning will be essential. Ultimately, in high-stakes AI, the wisdom to know when not to predict is as important as knowing what to predict.

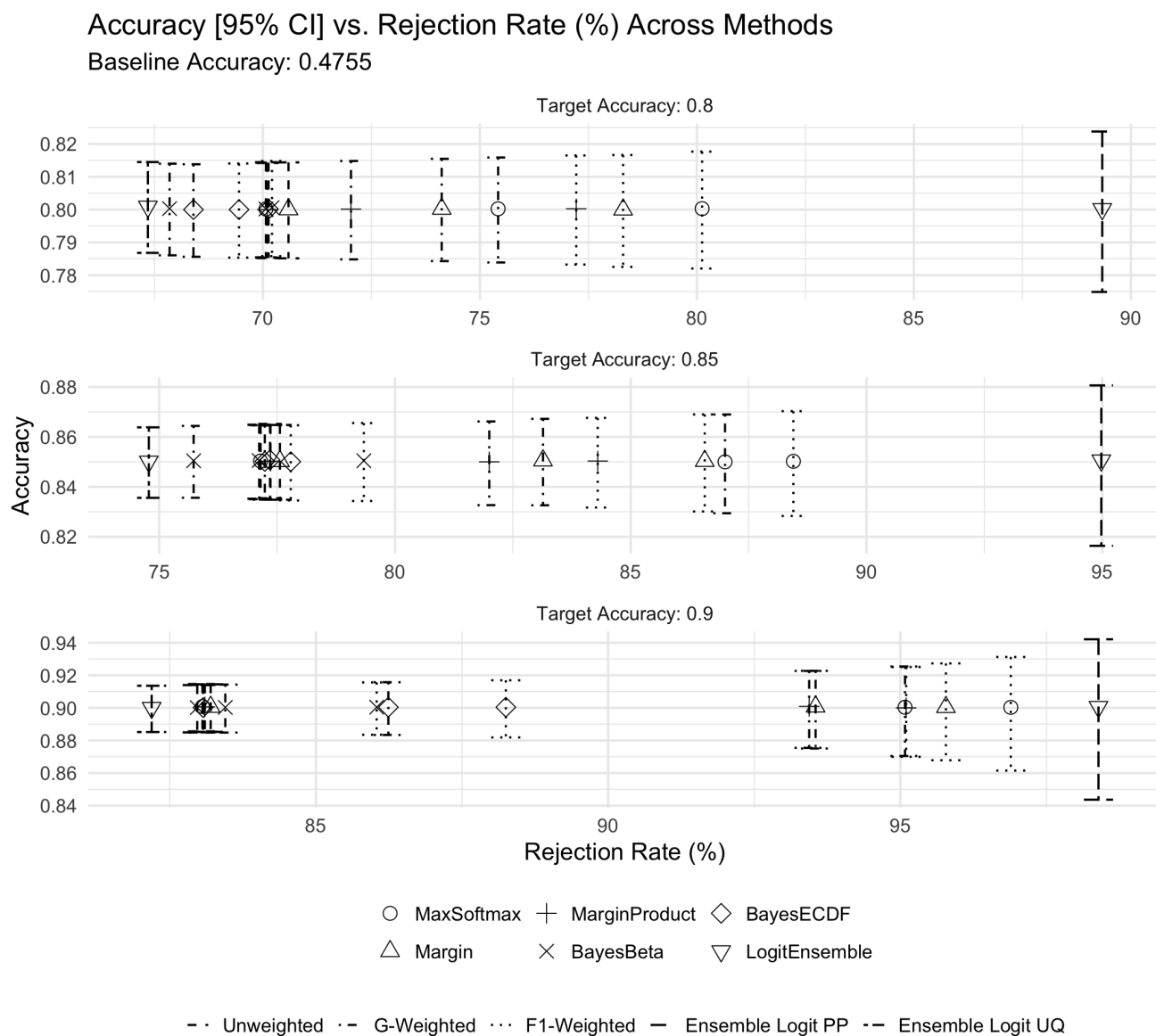


Figure 44.2: Selective accuracy–rejection trade-off on CIFAR-100 for all confidence scoring functions. For each method and target accuracy level, a threshold τ_α is chosen on the confidence score s , and predictions with $s < \tau_\alpha$ are rejected; the x-axis reports the resulting rejection rate $1 - \widehat{\text{coverage}}$ and the y-axis reports selective accuracy (accuracy computed on retained predictions only). Error bars denote 95% confidence intervals for the selective accuracy estimate on the retained set. Across targets, **LogitEnsemble-UQ** achieves the lowest rejection at comparable selective accuracy, indicating a more effective ranking of correct over incorrect predictions.

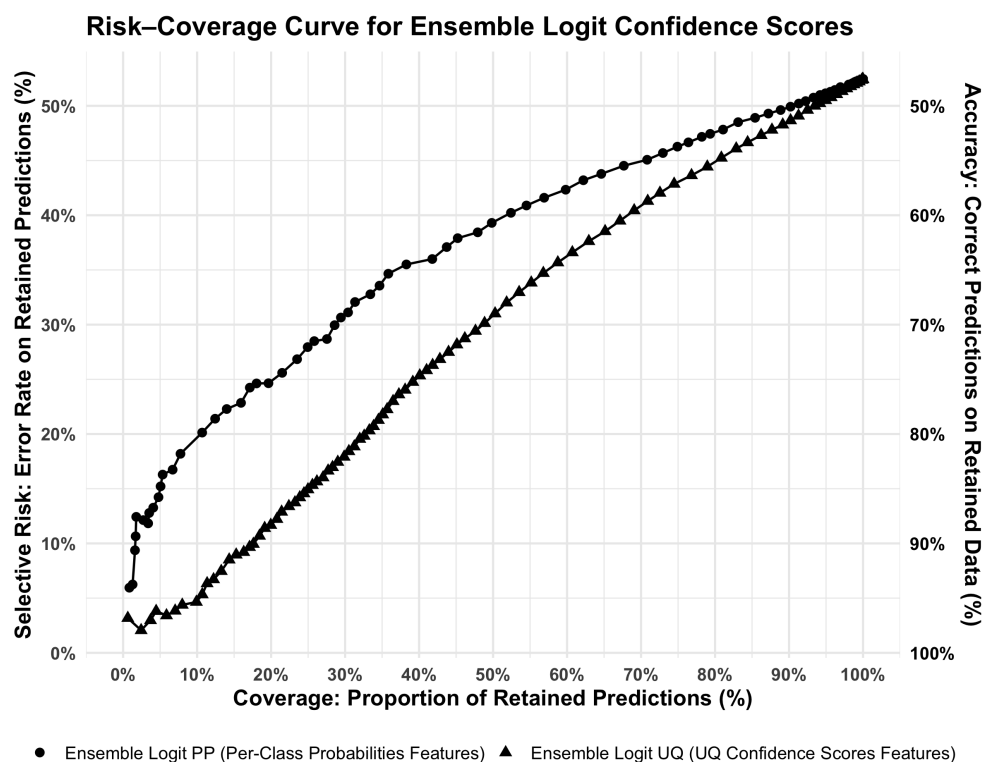


Figure 44.3: Risk-coverage curve for selective classification on CIFAR-100 using ensemble logistic-regression confidence scores. The x-axis shows *coverage* (percentage of predictions retained after applying a confidence threshold τ); the left y-axis shows *selective risk* (error rate computed on the retained predictions only), and the right y-axis shows the corresponding *selective accuracy* ($1 - \text{risk}$). Each point corresponds to a different threshold τ : moving to the left (lower coverage) uses a higher τ , rejects more low-confidence samples, and therefore typically decreases selective risk. Curves are shown for *Ensemble Logit PP* (features: per-class probabilities) and *Ensemble Logit UQ* (features: engineered uncertainty scores); the *Ensemble Logit UQ* curve lies below *Ensemble Logit PP*, indicating lower risk at the same coverage.

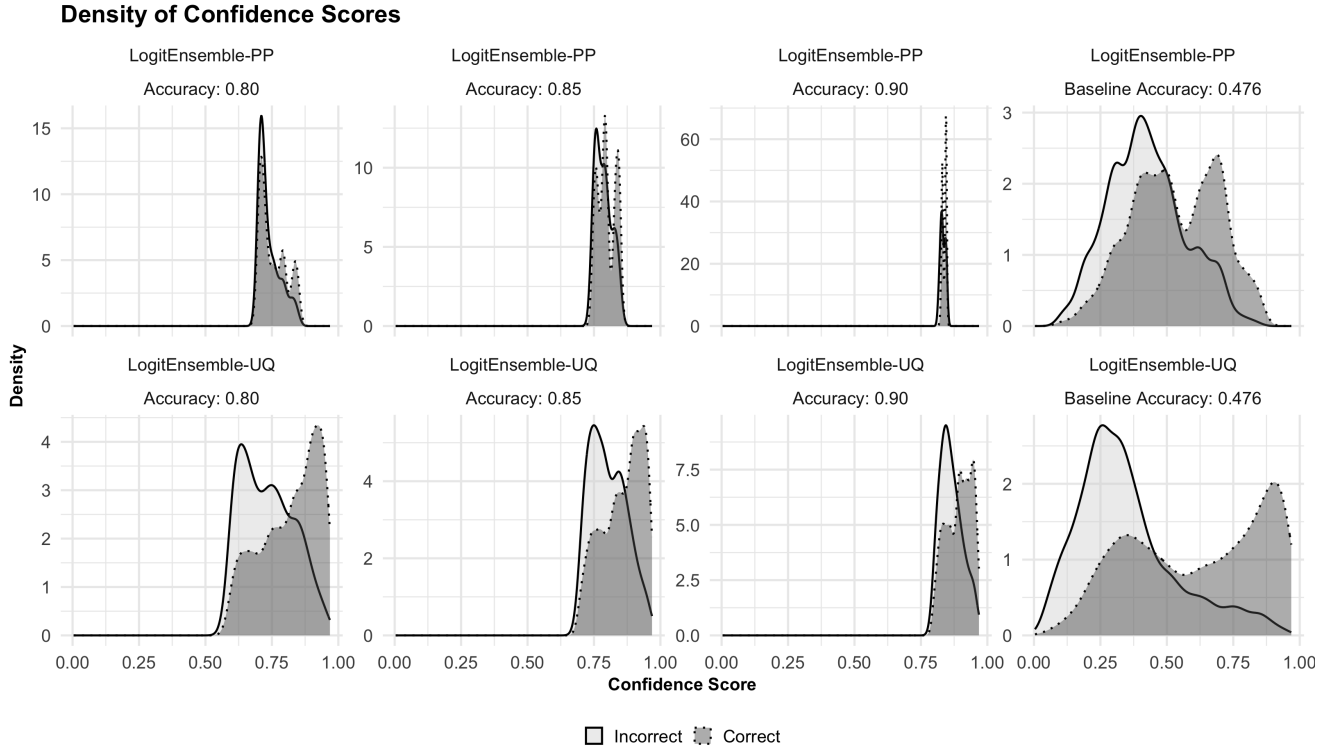


Figure 44.4: Confidence score density distributions for logistic regression ensembles at different operating points. For each target accuracy, we first select a threshold τ_α and *retain only* samples with confidence $s_i \geq \tau_\alpha$; the plotted curves therefore show the *conditional* densities of the retained set, $p(s | s \geq \tau_\alpha, \text{correct})$ and $p(s | s \geq \tau_\alpha, \text{incorrect})$. As the target accuracy increases, τ_α increases and the retained set becomes smaller, which truncates the left tail of the score distribution and changes the apparent densities across panels. Top panels correspond to *LogitEnsemble-PP*, which uses the raw predicted probability vector $\mathbf{p}_i$ as features. Bottom panels correspond to *LogitEnsemble-UQ*, which uses engineered uncertainty features s_i (including class-conditional weighting). The stronger separation between the retained correct and incorrect densities for *LogitEnsemble-UQ* indicates improved ranking of correct over incorrect predictions, yielding lower rejection for the same target accuracy.

Chapter 45

Biomedical data imputation

45.1 Introduction

Missing data persists as a major barrier in large-scale analyses of multivariate data, due to issues like incomplete collection, data availability, and low coverage. In biomedicine, missing data often arises in clinical trials, health surveys, or longitudinal studies, where patients may drop out or fail to answer certain questions due to various factors, such as side effects or worsening health conditions. Understanding the types of missingness—such as missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR)—is crucial for selecting appropriate analytical methods [1057]. Early approaches for dealing with missing data tend to reduce the generalisability of results or skew the trends present in the data [1519]. These include list-wise deletion, where only complete observations are considered, or imputations methods, where the complete data is used to predict plausible values for missing data points. Some of these imputation strategies include substitution by the mean of observed values, stochastic regression techniques, and hot-deck imputation. Single imputation approaches implicitly assume that the imputation is perfect and thereby fail to account for the uncertainty introduced by the prediction, while multiple imputation creates multiple copies of the data with different plausible values for the missing components.

More recently, machine learning approaches have been employed to deal with missingness, offering flexible solutions that can capture complex patterns in the data. For example, genomics datasets tend to be extremely high-dimensional, and deep learning approaches offer an attractive framework to capture high-dimensional relationships during imputation. In particular, deep generative models, such as variational autoencoders, offer an effective approach to learn joint distributions in high dimensions. This enables multiple imputation to account for uncertainty in missing data and subsequently greater robustness in downstream analysis.

45.2 Missing data

This section will explore the types of missing data, present biomedical examples, and briefly discuss traditional and machine learning methods used to address missing data.

45.2.1 Missing data types

Missing data can arise from a range of causes [369]:

- participants may withdraw from a study, or they could be removed if a treatment proves ineffective;
- people might choose not to respond to certain survey questions, such as ones related to gender or race;
- sometimes missingness is part of the study design. For example, limited resources may prevent follow-up with all subjects, leading to data gaps introduced through a known process, such as randomly selecting individuals to follow up.

In some situations, missing data can also occur for different reasons at the same time, and understanding why data is missing is critical for determining the best analysis approach. As stated in the work of Rubin [1432], the underlying cause of missingness plays a crucial role in determining whether a particular strategy is appropriate.

Missing data are therefore typically classified into three categories [1057, 1432], based on the underlying mechanism that leads to missing data.

Let $X = (x_{ij})$ denote a data set of size $N \times D$, where N represents the number of observations and D the number of variables. The i -th row of X , $x_i = (x_{i1}, \dots, x_{iD})$, contains the values for each variable corresponding to unit i , with x_{ij} representing the value of variable j for the i -th observation. In the case of missing data, it is useful to decompose the row x_i into two parts: x_i^{obs} and x_i^{mis} , which represent the observed and the missing values of x_i , respectively. To capture this missingness pattern, it is also convenient to introduce a *missingness indicator matrix* M , with elements

$$m_{ij} = \begin{cases} 1 & \text{if } x_{ij} \text{ is missing} \\ 0 & \text{if } x_{ij} \text{ is observed} \end{cases}. \quad (45.1)$$

Missing completely at random (MCAR). This is the simplest form of missingness and it occurs when the causes of the missingness are unrelated to the data. In this framework, for each observation, the probability of being missing does not depend on the values of any other variables in the data:

$$\mathbb{P}(m_i | x_i^{\text{obs}}, x_i^{\text{mis}}) = \mathbb{P}(m_i), \quad (45.2)$$

where m_i represents the vector of zeros and ones characterising the missing pattern. Examples of this missing scenario include situations such as the accidental drop or break of a blood sample tube from a participant, preventing the measurement of blood parameters, or the unintentional loss of a participant's questionnaire [578].

Missing at random (MAR). This assumes that the missingness mechanism can be explained by some of the observed features in the data set. Hence, for each observation x_i , the probability of missingness is conditionally independent of the unobserved data, given the observed data:

$$\mathbb{P}(m_i | x_i^{\text{obs}}, x_i^{\text{mis}}) = \mathbb{P}(m_i | x_i^{\text{obs}}). \quad (45.3)$$

Here, x_i^{obs} may also refer to a subset of the observed features. A classical example of this missing mechanism in clinical trials is when patients drop out based on (observable) characteristics, such as age or side effects of the treatment, but not on their (unobserved) final health status.

Missing not at random (MNAR). This occurs when the probability of missingness depends on the unobserved value itself (i.e. the missing feature), not just on the observed values:

$$\mathbb{P}(m_i | x_i^{\text{obs}}, x_i^{\text{mis}}) \neq \mathbb{P}(m_i | x_i^{\text{obs}}). \quad (45.4)$$

In the MNAR case, the missing variable cannot be predicted solely from the available data. If missing data are MNAR, valuable information is lost and there is no universally applicable method for properly addressing this type of missing data. For example, when a sensor fails to capture information beyond a certain threshold, its data are missing due to MNAR factors, and this is referred to as censoring [578].

Examples in biomedicine

Missing data is persistent in the field of biomedicine, spanning both clinical and experimental data settings. In clinical trials, for example, MNAR and MAR settings are frequent and can arise due to participant dropout before study completion or missed visits. These might originate from increased disease severity or even socio-demographic factors that might influence whether a participant will complete the study. Since missingness can be tied directly to important measurable outputs in these cases, imputation strategies that retain the integrity of the data and limit introduced biases are crucial for robust statistical analyses.

With the rapid expansion of high-throughput sequencing over the past few decades, a wealth of high-dimensional 'omics datasets has become available, offering novel insights into disease origins, progression, and opportunities for precision therapy [92]. However, these datasets are also prone to various forms of missingness. Depending on the modality – whether measuring gene expression (transcriptomics), protein abundance (proteomics), DNA sequence variation (genomics), genome accessibility (epigenomics), or metabolites (metabolomics) – each provides a distinct window into the underlying biological mechanisms, but each is also susceptible to missing data [797]. In transcriptomic data, for example, MCAR values may arise from random instrument failures, MAR values can occur when slight experimental differences are introduced across samples (e.g., use of different library preparation

kits), and MNAR values may result when low-abundance genes in a sample fall below the detection threshold of the sequencing platform.

Transcriptomic datasets can be measured in bulk, where gene expression is quantified across a group of cells (bulk RNA-sequencing; bulkRNA-seq), or at the single-cell level, where each cell is sequenced separately (single-cell RNA-sequencing; scRNA-seq). scRNA-seq is much more sparse and suffers from higher rates of missingness than bulkRNA-seq, where missingness rates are mostly due to real biological zeros, features of the experimental protocol, sequencing machine or the genes themselves [1352, 1800]. Missingness in bulkRNA-seq is relatively low compared to scRNA-seq, ranging from 10 – 40% [390, 1678], whereas scRNA-seq can be as high as 90% [429].

45.2.2 Missing data methods

Methods for handling missing data proposed in the literature can be broadly classified into several categories [578, 1057]:

Deletion. When certain variables contain missing values, a straightforward approach is to eliminate incomplete cases and only use observations with fully recorded data. This method, also referred to as *complete-case analysis* [1057], is simple to apply and may work well when missingness is minimal, but it can introduce significant bias, especially when the learning task involves inference on a subpopulation of the data.

Classic imputation methods. In these procedures, the missing values are estimated and replaced, allowing for the completed dataset to be analysed using standard techniques. The most common imputations methods include: *mean/median/mode imputation*, where the missing values are substituted with the mean (or median/mode) of the available observed data for each variable, and *hot-deck imputation* where missing data values are filled in by drawing from similar, observed data points that are considered comparable to the unit with missing data.

Imputation based on statistical machine learning methods. In this approach, missing data for an observation is predicted using a machine learning (ML) model, leveraging the relationships between observed variables. Some widely used ML methods for imputation include: *regression*, *k-nearest neighbours*, *random forests*, and more; they can be seen as extensions or algorithmic implementations of classical statistical imputation strategies. Although they may offer flexibility and scalability in high-dimensional settings, they still belong to the class of *single imputation* techniques. As such, they share a key limitation: imputed values are treated as *exact*, and therefore uncertainty in the imputation process is overlooked.

To address the uncertainty in imputation, *deep generative models* such as *Variational Autoencoders (VAEs)*, offer a more robust approach by modelling the underlying data distribution. VAEs can learn the conditional distribution of the missing data given the observed data and latent variables, enabling multiple imputations to be generated. This method accounts for uncertainty, as it provides a distribution over possible imputations rather than a single deterministic value.

45.3 Variational autoencoder

45.3.1 Deep latent variable models

Deep latent variable models (DLVMs), a specific class of generative models, aim to model the joint distribution of the observed variables, allowing for generation of new data points similar to the training data [895]. Generally, latent variable models assume that each data point—represented by x —is generated from a model depending on unobserved latent variables, denoted by $z \in \mathbb{R}^K$. These latent variables are typically assumed to lie in a much lower-dimensional space ($K \ll D$) and are used to explain the underlying structure of the observed data. DLVMs are widely used in many areas of ML, especially when dealing with complex or high-dimensional data.

In DLVMs, the joint density of the observed and latent variables is factorised as

$$p_{\theta}(x, z) = \underbrace{p(z)}_{\text{prior}} \underbrace{p_{\theta}(x | z)}_{\text{model}}. \quad (45.5)$$

The conditional density $p_{\theta}(x | z)$ employs deep learning (e.g. neural networks, with weights and biases contained in θ) to flexibly map the latent variables to parameters of the conditional density, while the prior $p(z)$ is typically assumed to be a standard normal distribution $\mathcal{N}(0, I)$, which dictates how the latent variables behave independently of the data.

There are often two aims in DLVMs: data generation and encoding. Generating a new data point x can be accomplished by sampling from

$$p_\theta(x) = \int p_\theta(x | z) p(z) dz. \quad (45.6)$$

Although this can be done by first sampling the latent variables from their prior, and then passing the sampled z through the deep learning model and generating x from its conditional distribution $p_\theta(x | z)$, this requires knowledge of the parameters θ . Ideally, the parameters θ would be learnt by optimising the intractable marginal likelihood

$$p_\theta(X) = \prod_{i=1}^N \int p_\theta(x_i | z) p(z) dz. \quad (45.7)$$

Encoding refers to learning the low-dimensional representation of the data, which amounts to computing the posterior of the latent variables given the observed data, defined as

$$p_\theta(z | x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x | z) p(z)}{\int p_\theta(x | z) p(z) dz}, \quad (45.8)$$

which again is intractable and requires knowledge of θ .

To overcome this intractability, VAEs employ *variational inference* to both approximate the posterior distribution and estimate θ . Instead of computing $p_\theta(z | x)$ directly, VAEs introduce a variational distribution $q_\phi(z | x)$, parameterised by a separate set of parameters ϕ , which serves as a tractable approximation to the true posterior, and find θ by introducing and optimising a lower bound to the marginal likelihood (45.7).

Diffusion models are a complementary family of latent variable generative models that can also be used for imputation; see Chapter 50 for a detailed discussion of this alternative approach and a comparison with VAEs.

45.3.2 Variational autoencoders

A variational autoencoder (VAE) is a particular DLVM [895], that employs neural networks to flexibly map the latent variables to the observed data and variational inference to address intractability. It is made up of two components:

- The *inference model*, also referred to as the *encoder* (or the *recognition model*) is the variational posterior of the form $q_\phi(z | x)$. It takes an observed data point, $x \in \mathbb{R}^D$, and encodes it in the lower-dimensional latent space, $z \in \mathbb{R}^K$, via the approximate posterior distribution of the latent variables $q_\phi(\cdot | x)$.
- The *generative model*, also called the *decoder*, of the form $p_\theta(x | z)$, takes latent variables z and reconstructs the data point, $\hat{x}$, via the generative model $p_\theta(\cdot | z)$.

While the prior on the latent variables $p(z)$ is often assumed to be a standard normal distribution $\mathcal{N}(0, I)$, the variational distribution $q_\phi(z | x)$ is a design decision that must be considered. Additionally, the choice of likelihood function $p_\theta(x | z)$ depends on the nature and statistical properties of the data, such as whether the observations are continuous or discrete. Below are some examples.

Example: Exponential family. Exponential family distributions are a class of probability distributions that can be used to model different types of data (e.g. real-valued, count, binary, and categorical). An *exponential family VAE* [424] chooses a generative model belonging to the exponential family, parametrised by a deep neural network:

$$p_\theta(x | z) = \text{ExpFam}(x; \eta_\theta(z)) \quad (45.9)$$

$$= \nu(x) \exp(\eta_\theta(z)^\top x - A(\eta_\theta(z))), \quad (45.10)$$

where:

- $\nu(x)$ is the base measure, which depends on the specific distribution in the exponential family;
- $\eta_\theta(z)$ is the natural parameter, which depends on the latent variable z , and specifically, is obtained as the output of the neural network with parameters θ and input z ;
- $A(\eta_\theta(z))$ is the log-normaliser of the exponential family.

This class includes several commonly used likelihoods, such as Gaussian, Poisson, and Bernoulli. Therefore, *exponential family VAEs* can be applied to a wide range of data types.

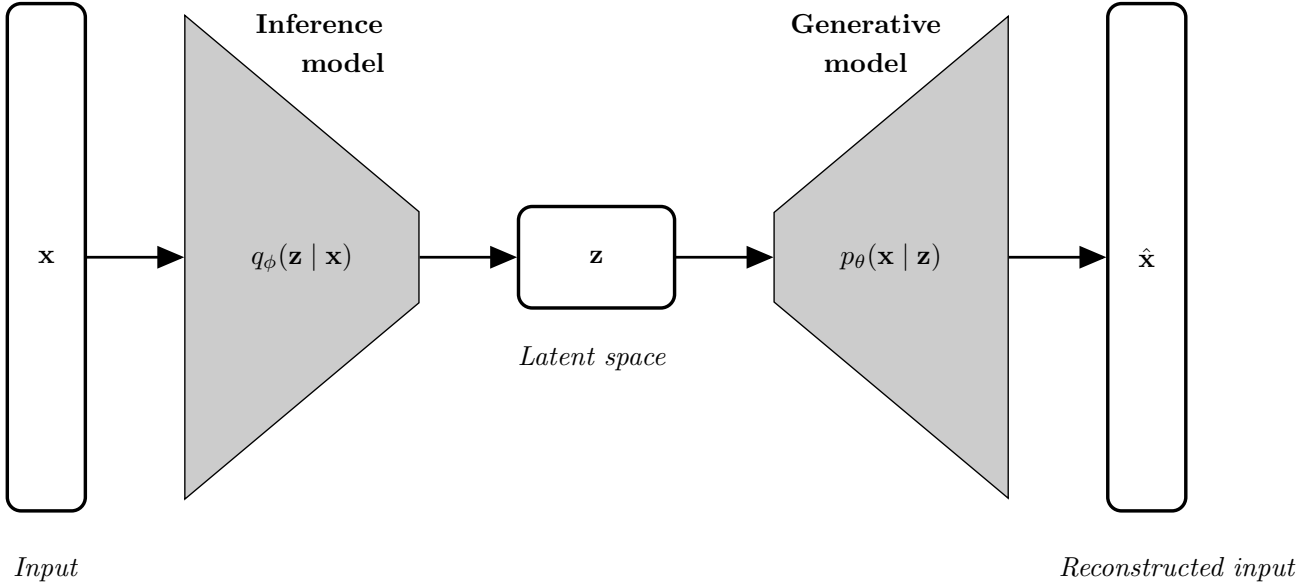


Figure 45.1: Structure of a variational autoencoder.

Example: Factorised Gaussian. In the case of continuous data, a standard choice for both the generative (decoder) and inference model (encoder) is a simple factorised Gaussian:

$$p_\theta(x | z) = \mathcal{N}(x; \mu_\theta(z), \text{diag}(\sigma_\theta^2(z))), \quad q_\phi(z | x) = \mathcal{N}(z; \mu_\phi(x), \text{diag}(\sigma_\phi^2(x))), \quad (45.11)$$

where $\mu_\theta(z) \in \mathbb{R}^D$, $\sigma_\theta^2(z) \in \mathbb{R}_+^D$, $\mu_\phi(x) \in \mathbb{R}^K$, $\sigma_\phi^2(x) \in \mathbb{R}_+^K$, and neural networks are employed to map the latent variables (or observed data) to the means and variances of the generative (or inference) model:

$$(\mu_\theta(z), \log[\sigma_\theta(z)]) = \text{DecoderNeuralNet}_\theta(z), \quad (45.12)$$

$$(\mu_\phi(x), \log[\sigma_\phi(x)]) = \text{EncoderNeuralNet}_\phi(x). \quad (45.13)$$

Biomedical application: In a biomedical setting, this choice is natural for gene expression data that has been z-scored, such that the expression of each gene across all observations has mean zero and unit standard deviation [895, 1352].

Example: Multivariate Gaussian. The factorised Gaussian (45.11) can be extended to a Gaussian with full covariance:

$$p_\theta(x | z) = \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z)), \quad q_\phi(z | x) = \mathcal{N}(z; \mu_\phi(x), \Sigma_\phi(x)). \quad (45.14)$$

In this case, the encoder and decoder networks are modified to output both the mean vectors and a parametrisation of the covariance matrices. A common choice is to use the Cholesky decomposition: $\Sigma_\theta(z) = L_\theta(z)L_\theta(z)^\top$ and $\Sigma_\phi(x) = L_\phi(x)L_\phi(x)^\top$, where $L_\theta(z)$ and $L_\phi(x)$ are lower triangular matrices, with non-zero elements given by the output of the networks. This approach guarantees positive semi-definiteness of the covariance matrices and facilitates stable training. For more details on the advantages of reparametrisation in VAEs, see [892].

Biomedical application: Multivariate Gaussians, combined with structured forms for the covariance (or precision) matrices to reduce dimensionality, might be applied to multi-omic data integration, whereby different 'omics modalities are modelled under a common framework [116].

Example: Negative binomial. Consider the case of discrete count data x , where $x \in \mathbb{N}^D$ may represent word counts, with D being the size of the vocabulary. A natural choice for the generative model (decoder) is the negative binomial [1865] defined by the shape/dispersion parameters $r \in \mathbb{R}_+^D$ and success probabilities $\rho \in (0, 1)^D$:

$$p_\theta(x | z) = \text{NB}(x; r_\theta(z), \rho_\theta(z)). \quad (45.15)$$

These parameters are derived from the latent variable z through nonlinear transformation defined by the decoder neural network:

$$(\log(r_\theta(z)), \text{sigmoid}^{-1}(\rho_\theta(z))) = \text{DecoderNeuralNet}_\theta(z). \quad (45.16)$$

This is referred to as the *negative binomial VAE* (NBVAE). In this model, the variational posterior (encoder) can be either a factorised or full Gaussian.

Biomedical application: scRNA-seq count data tends to be overdispersed and sparse, so is generally modelled using (zero-inflated) Poisson or negative binomial distributions. The NBVAE could therefore be a natural choice for this 'omics setting [673, 1074].

Example: Multinomial. An alternative method for modelling discrete data is to use a multinomial likelihood:

$$p_\theta(x | z) = \text{Multi}(x; \rho_\theta(z)), \quad (45.17)$$

where $\rho_\theta(z)$ lies in the simplex of dimension $D - 1$ and represents the probabilities of different categories (e.g., different words in a vocabulary). These probabilities are derived from the latent variable z via a neural network (decoder):

$$\text{softmax}^{-1}(\rho_\theta(z)) = \text{DecoderNeuralNet}_\theta(z). \quad (45.18)$$

Like NBVAEs, multinomial VAEs (MultiVAEs) [1170, 1865] use a factorised or full Gaussian variational posterior.

Remark 45.1. *It is worth noting that standard VAEs are generally not identifiable without additional restrictions on the model. This is not a major concern in cases where the primary goal is imputation rather than a scientifically meaningful representation of the latent space. When identifiability of the latent variables is of interest—for instance, when they are intended to carry a biological interpretation—identifiable yet flexible alternatives may be considered instead, such as structured factor models or constrained deep exponential-family models (see, e.g., [101, 1378]).*

Evidence lower bound (ELBO)

From a mathematical perspective, the goal is to find the model parameters θ (decoder) in order to maximise the marginal likelihood of the observed data, or equivalently $\log p_\theta(X)$, and derive the encoding of the data through the posterior $p_\theta(z | x)$. However, as discussed earlier, directly computing this likelihood and posterior is intractable. To address this, VAEs use a variational approximation of the posterior, $q_\phi(z | x)$, that leads to the introduction of the *evidence lower bound* (ELBO), which can be optimised as a proxy for maximising the marginal likelihood to find θ .

Starting from the log-marginal likelihood:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] = \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{p_\theta(z | x)} \right) \right] \quad (45.19)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z | x)} \frac{q_\phi(z | x)}{p_\theta(z | x)} \right) \right] \quad (45.20)$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z | x)} \right) \right]}_{\text{ELBO}_{\phi, \theta}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{q_\phi(z | x)}{p_\theta(z | x)} \right) \right]}_{\text{D}_{\text{KL}}(q_\phi(z|x) \| p_\theta(z|x))}, \quad (45.21)$$

we get the decomposition:

$$\log p_\theta(x) = \text{ELBO}_{\phi, \theta}(x) + \text{D}_{\text{KL}}(q_\phi(z | x) \| p_\theta(z | x)). \quad (45.22)$$

Since the KL divergence term is always non-negative, Equation (45.22) shows that maximising the ELBO is equivalent to maximising a lower bound on $\log p_\theta(x)$. By employing a flexible neural network to approximate the posterior distribution, we aim to minimise the KL divergence (obtaining a good approximation of the latent space), subsequently leading to a tight lower bound (better generative capabilities).

Optimising the ELBO

VAE aims to solve the optimisation problem:

$$\max_{\theta, \phi} \text{ELBO}_{\phi, \theta}(X) = \max_{\theta, \phi} \sum_{i=1}^N \text{ELBO}_{\phi, \theta}(x_i), \quad (45.23)$$

as the data points are assumed independent. In practice, it is useful to rewrite the objective function of our optimisation problem (45.23) in a form that highlights its two main components: the reconstruction loss and the

KL divergence. This decomposition allows us to better understand the objective we are minimising and how each part contributes to the overall training process.

For a data point x , the ELBO can be rewritten as:

$$\text{ELBO}_{\phi, \theta}(x) = \mathbb{E}_{q_{\phi}(z|x)} \left[\log \left(\frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)} \right) \right] \quad (45.24)$$

$$= \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + \mathbb{E}_{q_{\phi}(z|x)} \left[\log \left(\frac{p(z)}{q_{\phi}(z|x)} \right) \right] \quad (45.25)$$

$$= \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - \text{D}_{\text{KL}}(q_{\phi}(z|x) \| p(z)). \quad (45.26)$$

Hence, the optimisation problem in Equation 45.23 can be reformulated as the minimisation problem:

$$\min_{\theta, \phi} \underbrace{\sum_{i=1}^N \mathbb{E}_{q_{\phi}(z_i|x_i)} [-\log p_{\theta}(x_i|z_i)]}_{\text{Reconstruction Loss}} + \underbrace{\sum_{i=1}^N \text{D}_{\text{KL}}(q_{\phi}(z_i|x_i) \| p(z_i))}_{\text{KL Divergence}}, \quad (45.27)$$

balancing the trade-off between accurate data reconstruction and regularising the latent space according to the prior.

During training, the ELBO is maximised (or equivalently (45.27) is minimised) by stochastic gradient descent, which uses backpropagation through the hidden layers of the neural network to compute gradients and random sub-sampling of the data at each training step. To obtain an unbiased estimator of the required gradients, the reparametrisation trick [892, Section 2.4] is used, which applies a change of variables. For a fuller treatment of variational inference and the ELBO, see Chapter 12.

45.3.3 Beta-VAEs

In Bayesian statistics, the power likelihood is a generalisation of the classic likelihood that introduces an adjustable parameter to control the weight of the likelihood in the inference process. It offers a more flexible approach to learning from data and can be especially useful when dealing with noisy data, outliers or potential model misspecification [164, 772].

The power likelihood modifies the standard likelihood $p(x|\theta)$ by raising it to a power w ,

$$p_w(x|\theta) = p(x|\theta)^w, \quad (45.28)$$

with $w > 0$. In Bayesian inference, this likelihood is combined with the prior to form the posterior distribution:

$$p_w(\theta|x) \propto p(x|\theta)^w p(\theta). \quad (45.29)$$

Adjusting w provides a mechanism to control the balance between the prior and the likelihood: for $w < 1$, the influence of the data on the posterior is down-weighted, making the model more sensitive to the prior, while for $w > 1$ the likelihood is amplified, making the model more sensitive to the data.

For this reason, when the data contains outliers, the power likelihood with $w < 1$ reduces their influence and can lead to more robust inference, while using the standard likelihood can lead to over-fitting where the model tries to fit anomalous or unreliable data points. In the same way, when the model is misspecified—meaning the model may not perfectly match the true underlying data generating process—the full likelihood can drive the model to fit aspects of the data that are incorrectly captured by the misspecified model. Exploiting the power likelihood with a small w can mitigate this effect of misspecification giving more importance to the prior distribution.

In deep generative models, such as VAEs, given the complex and high-dimensional nature of the data, there is likely to be a mismatch between the model and the true data-generating distribution, which motivates the adoption of power likelihood in the learning process.

Power likelihood in ELBO

Motivated by this reasoning, we can apply the power likelihood in the ELBO in Equation 45.27. This leads to the following modified objective:

$$\sum_{i=1}^N \mathbb{E}_{q_{\phi}(z_i|x_i)} [-\log p_{\theta}(x_i|z_i)^w] + \text{D}_{\text{KL}}(q_{\phi}(z_i|x_i) \| p(z_i)), \quad (45.30)$$

where minimising (45.30) is equivalent to minimising the KL divergence between the variational posterior $q_\phi(Z | X)$ and the posterior under the power likelihood:

$$p_{\theta,w}(Z | X) \propto \prod_{i=1}^N p_\theta(x_i | z_i)^w p(z_i). \quad (45.31)$$

This naturally leads to the well-known β -VAEs [746], an extension on the classic VAE that includes a hyperparameter, referred to as β , that enforces regularisation on the latent loss.

In the ML community, β -VAEs are primarily known for improving disentanglement of the latent variables [271]. However, from a statistical perspective, their formulation can be understood as applying the power likelihood with $w = 1/\beta$, providing an alternative role of β in the model objective function. Indeed, it can be easily seen that the optimisation function used in training β -VAEs:

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(z_i|x_i)} [\log p_\theta(x_i | z_i)] - \beta \text{D}_{\text{KL}}(q_\phi(z_i | x_i) \| p(z_i)), \quad (45.32)$$

is equivalent to:

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(z_i|x_i)} \left[\log p_\theta(x_i | z_i)^{1/\beta} \right] - \text{D}_{\text{KL}}(q_\phi(z_i | x_i) \| p(z_i)), \quad (45.33)$$

which corresponds exactly to (45.30) with $w = 1/\beta$. In this framework, motivated by the use of power-likelihood, β controls the trade-off between the prior and the likelihood: for $\beta > 1$, the influence of the data on the posterior is down-weighted making the model more sensitive to the prior, while for $\beta < 1$ the likelihood is amplified, making the model more sensitive to the data. The classic VAE is retrieved for $\beta = 1$.

It is worth noticing that, after raising $p_\theta(x | z)$ to a power, the result is typically a generalized or tempered likelihood unless it is renormalized. In particular, in the factorized Gaussian case, this results in a covariance-inflation (or covariance deflation) interpretation briefly discussed in Section 45.4.1.

45.4 Variational autoencoders and missing data

The generative nature of DLVMs makes them particularly effective for data imputation tasks. They have become increasingly popular for handling missing data due to their ability to model complex, nonlinear relationships and intricate dependencies within multivariate datasets. Much of the progress in imputation using these models can be attributed to foundational generative models, such as VAEs.

One of the first applications of VAEs in missing data imputation appears in Rezende et al. [1391]. By introducing the reparametrisation trick, which enables efficient gradient-based optimisation for VAEs, they were able to achieve scalable training of deep generative models, setting the stage for their application in imputation. Building on this, Mattei and Frellsen [1146] proposed a modification that allows for exact conditional sampling in missing data scenarios. Following these advancements, Mattei and Frellsen [1147] introduced an extension that minimises a tighter lower bound based on importance weighted autoencoders [220]. To handle incomplete, heterogeneous data, Nazábal et al. [1225] developed an approach that incorporates both numerical and nominal data types for more complex imputation tasks. More recent methods, such as Ipsen et al. [820], continue to refine these models, especially in challenging settings such as MNAR. These developments highlight the continued progress in deep generative models for missing data imputation, with each new approach improving the model's ability to handle a wider variety of data types and missing data patterns.

Most approaches typically involve an initial imputation using a simple method (e.g., mean imputation) to create an initial complete dataset. Then, a VAE trained on this initial complete dataset provides parameters (ϕ, θ) . Assuming that each data point x_i can be partitioned into observed features x_i^{obs} and missing features x_i^{mis} , the goal is to use the trained VAE to estimate x_i^{mis} conditional on the observed components.

Single imputation

Single imputation provides a single estimate of the missing data. Under a squared error loss, the optimal imputation corresponds to the mean under the generative model, that is the conditional expectation

$$\hat{x}_i^{\text{mis}} = \mathbb{E}[x_i^{\text{mis}} | x_i^{\text{obs}}] = \int x_i^{\text{mis}} p_\theta(x_i^{\text{mis}} | x_i^{\text{obs}}) dx_i^{\text{mis}}. \quad (45.34)$$

where the conditional distribution is implicitly defined via the VAE's generative model. Since this distribution is generally intractable, it is typically approximated using an iterative scheme. In particular, noting that

$$\hat{x}_i^{\text{mis}} = \mathbb{E}[x_i^{\text{mis}} | x_i^{\text{obs}}] = \mathbb{E}[\mathbb{E}[x_i^{\text{mis}} | z_i, x_i^{\text{obs}}] | x_i^{\text{obs}}], \quad (45.35)$$

Qiu et al. [1352] iterate between an **encoding step**, which computes the expectation of z_i (mean of the encoder) given $\hat{x}_i^{\text{mis}}$ and x_i^{obs} :

$$\hat{z}_i = \mathbb{E}_{q_\phi(z_i | \hat{x}_i)}[z_i] = \int z_i q_\phi(z_i | \hat{x}_i^{\text{mis}}, x_i^{\text{obs}}) dz_i, \quad (45.36)$$

and a **decoding step**, which updates the imputed values via the decoder by computing the expectation of x_i^{mis} (mean of the decoder) given $\hat{z}_i$:

$$\hat{x}_i^{\text{mis}} = \mathbb{E}_{p_\theta(x_i^{\text{mis}} | x_i^{\text{obs}}, \hat{z}_i)}[x_i^{\text{mis}}] = \int x_i^{\text{mis}} p_\theta(x_i^{\text{mis}} | x_i^{\text{obs}}, \hat{z}_i) dx_i^{\text{mis}}. \quad (45.37)$$

These steps are repeated until convergence of the imputed values. Note that when the likelihood used in the decoder factorises across input dimensions (e.g., in a factorised Gaussian model), the conditional distribution $p_\theta(x_i^{\text{mis}} | x_i^{\text{obs}}, z_i)$ simplifies to $p_\theta(x_i^{\text{mis}} | z_i)$, further reducing the computational burden.

This procedure yields a single imputed value for each missing entry, hence the term single imputation. While effective and relatively straightforward, it does not account for uncertainty in the imputed values—a limitation addressed by multiple imputation strategies.

Multiple imputation

Multiple imputation [1057, 1215, 1519] explicitly incorporates the uncertainty of the missing data mechanism. The fundamental idea behind multiple imputation is to generate several completed datasets, each of which represents a different draw from the conditional distribution of the missing data given the observed data. These datasets are then analysed independently and final estimates are obtained by combining inference across all plausible datasets, for example by computing an overall mean and variance estimate for a given statistic. This approach allows the resulting inferences to reflect not only the uncertainty due to sampling variability, but also the additional uncertainty introduced by the imputation of missing values.

This methodology assumes that the data are MAR, where the probability of a value being missing depends only on the observed variables. Under this assumption, the goal is to obtain and sample from the predictive distribution of the missing data given the observed data, $p_\theta(x^{\text{mis}} | x^{\text{obs}})$, which takes the form:

$$p_\theta(x^{\text{mis}} | x^{\text{obs}}) = \int p_\theta(x^{\text{mis}}, z | x^{\text{obs}}) dz = \int p_\theta(x^{\text{mis}} | x^{\text{obs}}, z) p_\theta(z | x^{\text{obs}}) dz. \quad (45.38)$$

This expression highlights two sources of uncertainty: the variability in the latent parameters z , and the conditional variability in the missing values given z and the observed data.

While intractability of the posterior of the latent variables renders direct sampling infeasible, various approaches have been developed for approximate or asymptotically exact sampling, such as pseudo-Gibbs [1391] or Metropolis-within-Gibbs [1146]. Here, we focus on sample importance resampling [1147], and refer to Roskams-Hieter et al. [1422] for a description and comparison of these different sampling approaches. In sample importance resampling (SIR), the variational posterior (encoder) is used as the importance distribution. Specifically, for every data point i with missing features, samples of the latent variables $z_i^{(s)}$, for $s = 1, \dots, S$, are drawn from

$$z_i^{(s)} \sim q_\phi(z_i | x_i^{\text{obs}}, x_i^{\text{mis}(0)}), \quad (45.39)$$

where $x_i^{\text{mis}(0)}$ denotes an initial imputation (e.g. mean). The importance weights are proportional to

$$\omega_i^{(s)} \propto \frac{p_\theta(x_i^{\text{obs}} | z_i^{(s)}) p(z_i^{(s)})}{q_\phi(z_i^{(s)} | x_i^{\text{obs}}, x_i^{\text{mis}(0)})}. \quad (45.40)$$

Multiple imputations are obtained by first sampling the latent variables $z_i^{(k)}$, for $k = 1, \dots, K$, with replacement from the importance samples $z_i^{(s)}$, for $s = 1, \dots, S$, with weights $\omega_i^{(s)}$. Next, for each $z_i^{(k)}$, the missing data is imputed by sampling from:

$$x_i^{\text{mis}(k)} \sim p_\theta(x_i^{\text{mis}} | z_i^{(k)}, x_i^{\text{obs}}). \quad (45.41)$$

In contrast to Gibbs, an advantage of SIR is parallelisability. To mitigate degeneracy of the weights, a heavy tailed Student-t proposal is recommended, with mean and variance given by the encoder.

It is worth noticing that the Multiple SIR-based imputations mainly capture predictive uncertainty conditional on the fitted VAE parameters. They do not capture full parameter uncertainty unless one adds posterior sampling, bootstrap, or a related approximation.

45.4.1 Multiple imputation with beta-VAEs

Although expressive and powerful, deep models have been shown to be overconfident even when predictions are incorrect [1582] and also underestimate the variability of out-of-sample test data [1235]. In the context of multiple imputation, this leads to underestimation of the uncertainty in missing data [1422]. To combat this, β -VAEs have been employed, which make use of the power likelihood to avoid overfitting and achieve well-calibrated uncertainty of the missing data.

For multiple imputation with a β -VAE, the predictive distribution (in Equation 45.38) is constructed from the power likelihood, that is, the likelihood of the generative model is raised to the power $1/\beta$ (see Equation (45.33)):

$$p_{\theta,\beta}(x^{\text{mis}} | x^{\text{obs}}) \propto \int p_{\theta}(x^{\text{mis}} | x^{\text{obs}}, z)^{1/\beta} p_{\theta,\beta}(z | x^{\text{obs}}) dz. \quad (45.42)$$

It can be noticed that in the case of the factored Gaussian generative model, the power likelihood $p_{\theta}(x^{\text{mis}} | x^{\text{obs}}, z)^{1/\beta}$ is simply proportional to a Gaussian with variance rescaled by a factor of β . On the other hand, $p_{\theta,\beta}(z | x^{\text{obs}})$ represents the intractable true posterior of the latent variables under the power likelihood given the observed data only, and to sample from it, the SIR algorithm can be adjusted accordingly.

45.4.2 Tuning beta

The β parameter in a β -VAE plays a crucial role in controlling the strength of the regularisation imposed by the KL divergence term, thereby influencing the trade-off between accurate imputation and uncertainty quantification in the missing values. Choosing an appropriate value for β is therefore essential. In principle, many approaches could be used to tune β , depending on the modelling goals and constraints. Below we discuss two strategies: one based on evaluating predictive performance, and another grounded in Bayesian model selection. In both cases, a subset of the observed data is masked, in order to provide ground truth values of the missing data. The β -VAE is trained on this subset of data, for a range of different choices of β ; the strategies below are used to select β among these choices, and a β -VAE is subsequently retrained on the original data with the selected β .

Tuning via predictive performance

A common approach is to evaluate model performance using metrics such as the mean absolute error (MAE) to assess imputation accuracy and empirical coverage (EC) to quantify uncertainty in the missing values. The **mean absolute error** measures the average magnitude of the errors between the imputed values and the truth (for the subset of masked data). Formally, it is defined as:

$$\text{MAE} = \frac{1}{N^{\text{mis}}} \sum_{i=1}^{N^{\text{mis}}} \|\hat{x}_i^{\text{mis}} - x_i^{\text{mis}}\|_1, \quad (45.43)$$

where $\|\cdot\|_1$ represents the L_1 distance and N^{mis} is the total number of masked values. Note that in multiple imputation, imputed values $\hat{x}_i^{\text{mis}}$ are obtained as an average across the K imputed datasets:

$$\hat{x}_i^{\text{mis}} = \frac{1}{K} \sum_{k=1}^K x_i^{\text{mis}(k)}. \quad (45.44)$$

Empirical coverage evaluates the proportion of true values that fall within a $100(1-\alpha)\%$ confidence interval (CI). For instance, given a significance level α , the pointwise CIs, denoted by $[l_i, u_i]$, containing the lower and upper bounds for the missing values of the i th point, are computed from quantiles of the K imputed values. Then, the EC is computed as:

$$\text{EC}(\alpha) = \frac{1}{N^{\text{mis}}} \sum_{i=1}^{N^{\text{mis}}} \sum_{j=1}^D m_{ij} \mathbb{1}(x_{ij} \in [l_{ij}, u_{ij}]). \quad (45.45)$$

The goal is to select the β that best balances these two objectives: minimising MAE while maximising EC. For this reason, instead of looking separately at the MAE and EC, one may also want to consider a trade-off measure such as:

$$\text{MAE} + \gamma \max(0, 1 - \alpha - \text{EC}(\alpha)) , \quad (45.46)$$

that needs to be minimised (where $\gamma > 0$). The metric introduced in Equation (45.46) can be interpreted as a penalized version of the MAE where the hyperparameter γ controls how strongly we penalize models with too small empirical coverage. This provides a convenient way to summarize the accuracy-coverage trade-off in a single criterion, although it comes at the cost of having to set γ .

Tuning via marginal likelihood

A more principled, fully Bayesian alternative is to select β by maximising the marginal likelihood of the missing data given the observed data, that is

$$p_{\theta,\beta}(x^{\text{mis}} | x^{\text{obs}}) = \prod_{i=1}^{N^{\text{mis}}} p_{\theta,\beta}(x_i^{\text{mis}} | x_i^{\text{obs}}) . \quad (45.47)$$

For each data point, an unbiased approximation of the quantity $p_{\theta,\beta}(x_i^{\text{mis}} | x_i^{\text{obs}})$ is obtained from the importance sampler as:

$$\widehat{p_{\theta,\beta}}(x_i^{\text{mis}} | x_i^{\text{obs}}) = \sum_{s=1}^S \omega_i^{(s)} p_{\theta,\beta}(x_i^{\text{mis}} | x_i^{\text{obs}}, z_i^{(s)}) . \quad (45.48)$$

45.5 Biomedical application

This section illustrates how VAEs and their generalisations (β -VAEs) can be applied to missing data problems in biomedical settings. We focus on evaluating both the quality of the imputations and their impact on downstream analysis tasks. The experiments are conducted on real genomic data, where missingness is introduced under various controlled scenarios. We begin by describing the dataset and the missingness mechanisms. Next, the core results are presented, highlighting the differences between standard VAEs and β -VAEs in terms of uncertainty quantification and the impact on downstream tasks.

45.5.1 Dataset and missingness scenarios

We focus on high-dimensional transcriptomic cancer datasets from the Cancer Genome Atlas (TCGA) that quantify gene expression across 17,175 genes and hundreds of patient samples. Each observation represents a patient who has been diagnosed with either glioblastoma (GBM; $N = 159$), lower-grade glioblastoma (LGG; $N = 508$), lung adenocarcinoma (LUAD; $N = 478$) or lung squamous carcinoma (LUSC; $N = 475$). Missing data are introduced under different scenarios, including MCAR and MNAR, and imputation performance is evaluated by assessing the accuracy of imputation (MAE), coverage (EC), and false positive rates (FPR) of downstream tasks. The MCAR setting was generated by randomly masking 5%, 10% and 20% of the values in 20% of the observations across both pairs of cancer types, paired by tissue type (LGG/GBM and LUAD/LUSC). The MNAR setting was designed to mimic a real MNAR setting that occurs in transcriptomic datasets, whereby genes that have high GC content are more likely to experience dropout during sequencing [1352]. In this case, half of the genes with the top 5%, 10% and 20% highest GC content were randomly masked in 20% of the observations in the original transcriptomic datasets to mimic this scenario. This procedure generated 6 missing scenarios across two types (MCAR and MNAR) and three levels (5%, 10% and 20%) for each paired dataset, LGG/GBM and LUAD/LUSC.

45.5.2 Experiment design

After simulating missingness as described above, the datasets are scaled and the VAEs are subsequently trained using zero imputation at missing value indices. The same model and hyper-parameters described in [1422] are adopted, specifically 250 training epochs, a learning rate of 10^{-5} , two hidden layers of dimension 600, 2000, and a latent space dimension of 200. The grid of β values investigated ranges from 0.4 to 3, and β is tuned using the metric defined in Equation (45.46) with $\gamma = 1$. Once the models are trained, they are used to impute $K = 100$ datasets. Code to reproduce all the experiments is available at <https://github.com/roskamsh/BDL-experiments>.

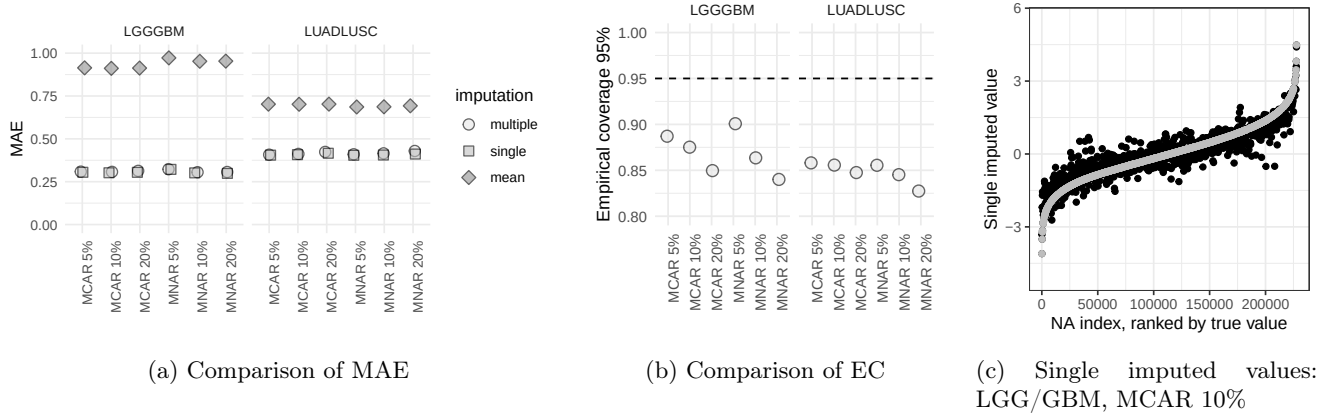


Figure 45.2: Standard VAE ($\beta = 1$) underestimates the uncertainty in multiple imputation.

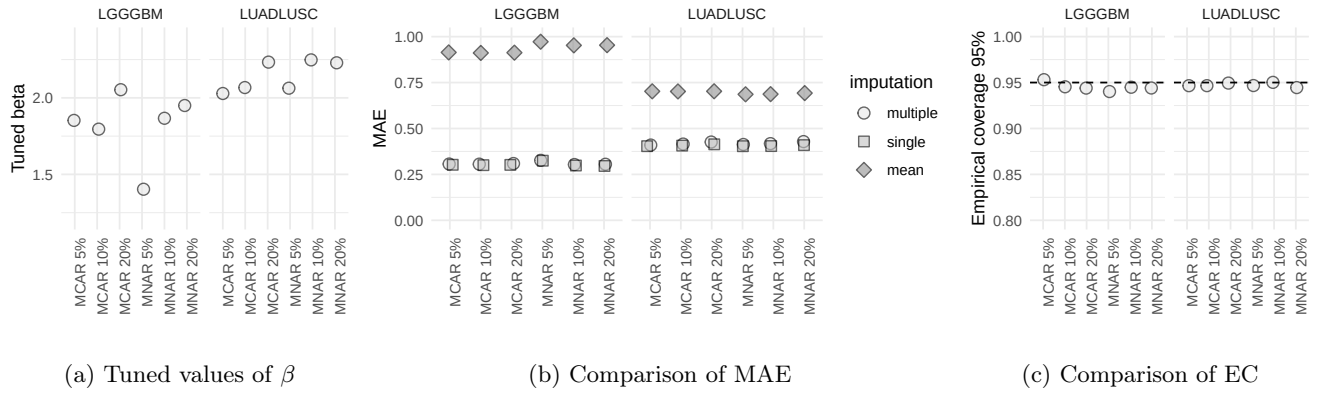


Figure 45.3: Multiple imputation with β -VAEs. (a) Selected β values tend to increase with the proportion of missingness, (b) MAE results are comparable, yet (c) EC improves significantly over standard VAEs.

45.5.3 Experimental results

This section illustrates how VAEs and β -VAEs can be used for uncertainty-aware imputation of missing data. Three key observations emerge from the experiments.

Overconfident imputations in standard VAEs

Standard VAEs tend to produce overconfident imputations when used for multiple imputation. The variability derived by sampling from the posterior over latent space is often insufficient. This results in EC—the proportion of masked values falling within the nominal 95% CIs—well below nominal confidence levels, especially for more extreme or outlying values (see Figure 45.2c, which depicts true values (gray) against true ranked indices). This behaviour is illustrated in Figure 45.2, where MAE is comparable between single and multiple imputations (Figure 45.2a), yet EC is too low (Figure 45.2b), indicating overconfidence, especially as the level of missingness increases.

Improved uncertainty quantification via beta-VAEs

The introduction of β -VAEs improves the quantification of uncertainty in the imputations. Introducing a tunable β parameter in the ELBO formulation enables control over the trade-off between imputation accuracy and uncertainty. By increasing the weight on the KL divergence term in the ELBO, these models encourage posterior distributions that are closer to the prior, resulting in more dispersed latent representations. Figure 45.3 shows that the imputation accuracy (measured by MAE) is similar to standard VAEs (Figure 45.3b), yet β -VAEs achieve coverage levels close to the target 95% (Figure 45.3c), correcting the underestimation observed in standard VAEs. It is worth noting that the optimal β values (Figure 45.3a) tend to increase as the proportion of missing data grows, suggesting that stronger regularisation and greater prior influence is needed under higher uncertainty.

Effects on downstream tasks

The quality of imputation—both in terms of accuracy and uncertainty—has a substantial impact on downstream tasks. To illustrate this, a logistic regression model with LASSO penalty was used to identify gene sets discriminating between cancer types. The model was fit in four data settings: the fully observed ground truth data with no missingness, the single dataset with mean imputation, the single dataset with standard VAE imputation, and each of the 100 datasets generated via multiple imputations using the tuned β -VAEs.

Model coefficients were compared in terms of MAE against the ground truth estimates. As shown in Figure 45.4, naive strategies, such as mean imputation, result in poor performance with high MAE and hence unreliable coefficient estimates. The flexibility afforded by VAEs helps to combat this, and multiple imputation with β -VAEs achieves slightly better performances compared to standard VAEs, in certain cases such as in MCAR contexts with high missingness (20%). Additionally, multiple imputation allows construction of CIs that reflect variability in the coefficients due to the missing data; the 95% EC of the coefficients lies close to the desired level of 0.95.

A key goal in biomedical inference is the identification of stable and interpretable features, such as genes associated with cancer subtypes. In this context, the sparse coefficient estimates obtained through the LASSO penalty identify discriminating genes. For multiple imputations, a gene is selected if its inclusion probability—defined as the proportion of imputations for which its LASSO coefficient is non-zero—exceeds a given threshold. False positive rates (FPR) and true positive rates (TPR) are computed according to the discriminating gene sets identified in the “true” complete dataset (Figure 45.5). Naive mean imputation yields inflated FPRs, low TPRs, and unreliable selection results, consistent with its poor performance in coefficient estimation (Figure 45.4). Again, VAEs allow for improved gene selection, with multiple imputation leading to lower false discovery of genes in various cases (notably MNAR settings for LGG/GBM in Figure 45.5a). Hence, multiple imputation with β -VAEs not only provides well-calibrated uncertainty in imputations but also results in more acceptable FPRs in downstream gene discovery.

45.6 Discussion

Deep latent variable models offer an attractive framework to capture complex patterns in high-dimensional data. In biomedicine, where large numbers of variables are typically recorded and missingness is commonplace, DLVMs are useful to deal with missing data through imputation. Compared to simple strategies, such as mean imputation, they offer better imputation accuracy, leading to substantial improvements in downstream analysis, including coefficient estimation and variable selection. Moreover, their generative nature allows multiple imputations to account for uncertainty in missing values. This helps to improve robustness in downstream tasks, such as lower false discovery of discriminating genes.

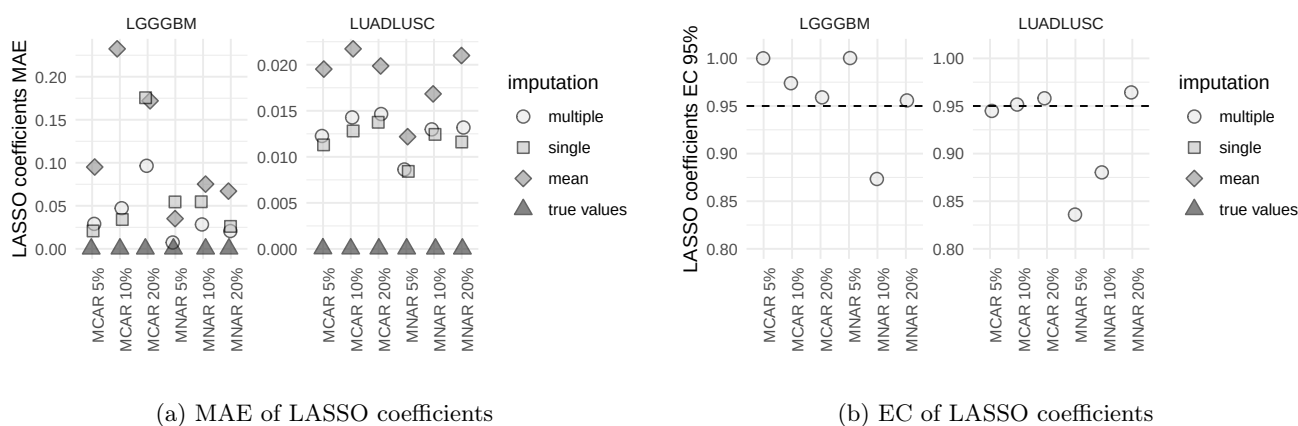


Figure 45.4: LASSO coefficients compared between ground truth and imputed datasets. Multiple imputation tends to improve accuracy and provide good empirical coverage.

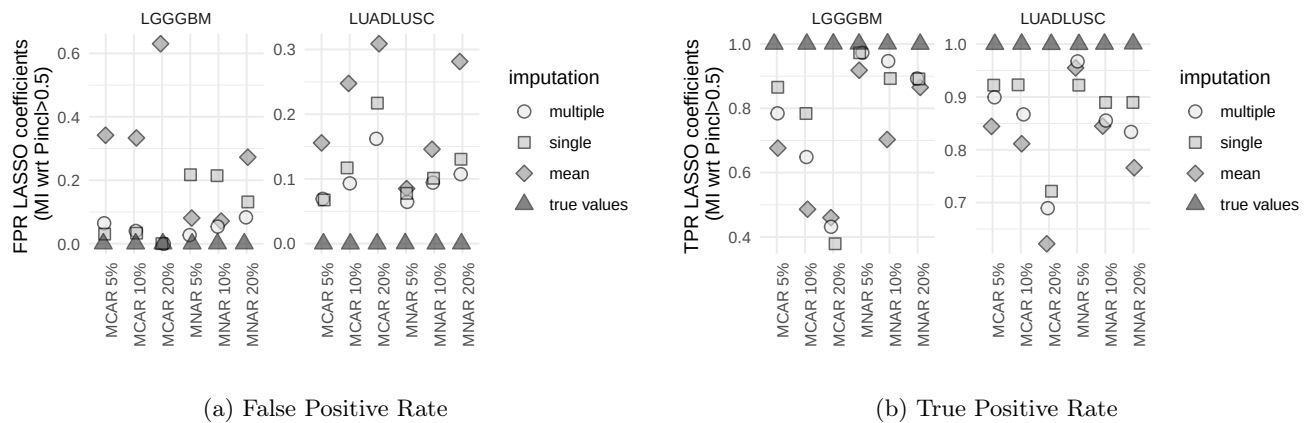


Figure 45.5: FPR and TPR in gene selection. Uncertainty-aware imputation reduces FPRs, particularly under MNAR. At the same time, TPR is similar, with some improvements or reductions in different settings.

Chapter 46

Deep Bayesian spatio-temporal modelling of wildfires

Wildfires present an ever-present threat to human life, property, health, and the ecosystem. Wildfires are increasing in size and intensity due to climate change, increase in fuel loads, changing land management practices, and an expanding wildland-urban interface. Indeed, in recent years we have seen examples of unusually destructive wildfires across the world, such as those in Australia (2023-24), Canada (2023), Europe (2023), South America (2024), and Southern California (2025). There is a great need to model various aspects of wildfire to learn about and mitigate risk, health impacts, and spread after ignition. This can be challenging because wildfires are complex, representing interacting spatio-temporal processes across many scales and drivers. Such complex processes require models that can accommodate complicated dependence structure and that adequately quantify uncertainty across data, process, and parameters. The deep Bayesian paradigm is ideally suited for these tasks.

46.1 Introduction

Wildfires represent a complex chaotic system driven by extreme weather conditions, variable fuel loads, complex terrain, and physical and biological processes involving feedback across multiple scales in space and time. This, combined with human behaviour (management, mitigation, and accidental or deliberate ignition) and limited data availability, make this one of our most challenging environmental modelling problems.

Since the mid-1990s, statisticians have approached the modelling of complex spatio-temporal dynamical systems via a multi-level Bayesian hierarchical model (BHM) paradigm. In 1996, Berliner [127] described a general hierarchical model for such processes that focused on modelling the process and parameters given observations (i.e., the posterior distribution) as proportional to the product of conditional distributions (data given process and parameters, process given parameters, and prior distributions on parameters). Complex processes require each of these component distributions to be decomposed into sub-models, making the BHM very much a “deep” model (e.g., multiple data sets conditioned on a common latent process, process parameters conditioned on covariates, Gaussian processes, or Markov random fields, etc.; see [1753] and the overview in [345]). One of the strengths of this paradigm is the ability to incorporate prior mechanistic understanding of the process (e.g., motivating the space-time dynamic process via a discretised partial differential equation, incorporating deterministic model output as data, or informing prior distributions from deterministic model output; see [1008, 1750, 1754]).

In recent years, there has been more interest in the development of hybrid statistical/deep neural models that can incorporate the expressiveness of deep neural models, yet can also include dependent random effects, measurement uncertainty, prior information, and missing data in a more formal context than in traditional deep neural models (see the discussion in [1751, 1752]). Examples include the use of convolutional neural networks to learn the transition operator in an integro-difference equation spatio-temporal dynamic model [1830], the incorporation of a deep echo state network within a mechanistic-informed process model [1818], and the incorporation of a novel statistical model for spatio-temporal extremes within a variational autoencoder [1850, 1851]. We present these latter two examples in this chapter. These hybrid models (embedding a deep neural model in a statistical model, or embedding a statistical model in a deep learning framework) leverage the flexibility and generalisability of the deep neural framework and the formal uncertainty quantification components of Bayesian statistical modelling. These approaches tend to work well when the overall framework is a BHM, while incorporating the spirit of deep neural uncertainty quantification that is the foundation of Bayesian deep learning as exemplified in this book.

Bayesian models for probabilistic estimation of fire risk started becoming prevalent in the literature in the early 2000s [207, 1762] and Bayesian networks started being used for evaluating fire risk in the early 2010s [431]. Around the same time, Bayesian methods combined with extreme value theory were used for fire size prediction [1163]. Extreme value theory has remained important for modelling fire risk [853, 910, 983, 1326] and served as a data challenge for the Extreme Value Analysis conference in 2021 [1263]. More recently, there has been a growing interest in modelling wildfire spread with Bayesian methods, some examples of which also considered deep neural models [201, 669, 670, 1568, 1819].

In the next two sections we focus on two examples, one that considers a specialized variational autoencoder to model dependent extremes in spatio-temporal data for a fire weather index, and the other that considers a data-driven level set/deep echo state network model to forecast the spread of a fire front.

46.2 Application: Analysis of Extremes in Fire Weather Index with an Extremes-Aware Variational Autoencoder

Environmental extremes such as wildfires typically appear in clusters in space and time. Traditionally, it has been difficult to model such dependent extremes in high dimensions. Gaussian processes, which are quite useful for modelling the bulk of many environmental spatial and spatio-temporal processes (either directly or as a latent process) are not appropriate given that the dependence in the tails between any two locations, no matter how close, are asymptotically independent (the extremal dependence is guaranteed to go to zero as the quantile increases). More classic distributions for extremes (e.g., generalized extreme value distributions) have been quite useful for looking at return times of extreme events for one location in space, but are more difficult to model joint dependence. Over the last two decades, innovative models for dependent extremes have been constructed, but are often not practical for flexible types of dependence (e.g., max-stable processes only allow asymptotic dependence no matter how far apart). In addition, these approaches have proven to be difficult to scale to very large numbers of spatial locations; see [805] for an overview of recent developments in spatial extremes.

A flexible spatio-temporal extremes model for environmental processes should allow: (a) close locations to have stronger dependence than locations far apart; (b) dependence strength to decay as “extremeness” intensifies; (c) dependence structure to be heterogeneous (non-stationary) across large domains (e.g., continental North America); and (d) both long-range asymptotic independence in the tails and short-range asymptotic independence or dependence to be present. Based on these criteria, [1851] extended the so-called “max-ID” extremes model of [188, 1386] to give a model that exhibits flexible tail characteristics across the domain and does not impose a dependence class *a priori*. As described below in Section 46.2.1, this model is represented as a Bayesian hierarchical model, but its calculation by traditional Bayesian computation approaches (Markov chain Monte Carlo or Hamiltonian Monte Carlo) is prohibitive on large spatial domains. Thus, [1851] embed their novel extremes model within a variational autoencoder, as described below in Section 46.2.2. This model has been used for uncertainty quantification over regions of extreme sea surface temperature in the Red Sea that can lead to coral bleaching [1851] as well as emulating the turbulent plume associated with a computational intensive fluid dynamics model of wildfire [1850].

46.2.1 Flexible Max-ID Extremes Model

As in [1851], we factor our spatio-temporal data of interest as

$$X_t(\xi) = \epsilon_t(\xi)H_t(\xi), \quad \xi \in \mathcal{S}, \quad (46.1)$$

where $t = 1, \dots, T$ indexes time, $\mathcal{S} \subset \mathbb{R}^2$ is the spatial domain of interest and $\epsilon_t(\xi)$ is a noise process with independent Fréchet($0, \tau, 1/\alpha_0$) marginal distributions; that is, $\Pr\{\epsilon_t(\xi) \leq x\} = \exp\{-(x/\tau)^{-1/\alpha_0}\}$, where $x > 0$, $\tau > 0$ and $\alpha_0 > 0$. Then, $H_t(\xi)$ is constructed using a low-rank representation:

$$H_t(\xi) = \left\{ \sum_{k=1}^K \omega_k(\xi)^{1/\alpha} Z_{kt} \right\}^{\alpha_0}, \quad (46.2)$$

where $\alpha \in (0, 1)$, $\{\omega_k(\xi) : k = 1, \dots, K\}$ are fixed compactly-supported radial basis functions centred at K pre-specified knots such that $\sum_{k=1}^K \omega_k(\xi) = 1$ for any $\xi \in \mathcal{S}$, and $\mathbf{Z}_t = \{Z_{kt} : k = 1, \dots, K\}$ are independently distributed as exponentially-tilted positive-stable (expPS) random variables, whose densities are of the form

$$h(x; \alpha, \gamma_k) = \frac{f_\alpha(x) \exp(-\gamma_k x)}{\exp(-\gamma_k^\alpha)}, \quad x > 0, \quad k = 1, \dots, K. \quad (46.3)$$

Here, f_α is the density function of a positive-stable variable defined through its Laplace transform $\int_{\mathbb{R}^+} \exp(-rx) f_\alpha(x) dx = \exp(-r^\alpha)$, $r \geq 0$, $\alpha \in (0, 1)$ determines the rate at which the power-law tail of f_α tapers off, and the tilting parameters $\gamma_k \geq 0$ determine the extent of tilting, with larger values of γ_k leading to lighter-tailed Z_{kt} .

Given a finite set of observation locations $\{\xi_1, \dots, \xi_m\}$, we define the discretised spatio-temporal process as $\mathbf{X}_t = \{X_t(\xi_i) : i = 1, \dots, m\}$. To implement the Bayesian hierarchical model, we aim to estimate both the model parameters $\phi = (\alpha, \tau, \alpha_0, \gamma_1, \dots, \gamma_K)^\top$ and the latent process given by $\{\mathbf{Z}_t : t = 1, \dots, T\}$. Full Bayesian inference (e.g., via Markov chain Monte Carlo) becomes computationally prohibitive due to the high dimensionality of the parameter space and the latent process.

46.2.2 Extremes-Aware Variational Autoencoder (xVAE)

A variational autoencoder (VAE) is a type of generative model that learns to represent data in a lower-dimensional latent space through an encoder (inference network) and generate new data similar to what it has seen during training through a decoder (generative network) [892]. Utilizing the hierarchical construct of our model described in (46.1)–(46.2), we use the encoder to learn to map input data $\mathbf{X}_t$ to a distribution over latent variables $\mathbf{Z}_t$ in encoded space:

$$\begin{aligned} q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t) &= \mathcal{N}(\boldsymbol{\mu}_t, \text{diag}(\boldsymbol{\sigma}_t^2)), \\ (\boldsymbol{\mu}_t^\top, \log \boldsymbol{\sigma}_t^\top)^\top &= \text{EncoderNeuralNet}_{\theta_e}(\mathbf{x}_t), \end{aligned} \quad (46.4)$$

where θ_e are the bias and weight parameters of the encoder neural network. The decoder then samples from q_{θ_e} and learns to map sampled latent variables $\mathbf{z}_t$ back to reconstructed data using the distribution $p_\phi(\mathbf{x}_t | \mathbf{z}_t)$ defined by (46.1)–(46.2). Critically, the model parameters $\phi = (\alpha, \tau, \alpha_0, \gamma_1, \dots, \gamma_K)^\top$ are estimated by the decoder network:

$$\hat{\phi} = \text{DecoderNeuralNet}_{\theta_d}(\mathbf{z}_1, \dots, \mathbf{z}_T), \quad (46.5)$$

where θ_d are the bias and weight parameters of the decoder neural network, and positivity of the parameter estimates is enforced through appropriate output-layer transformation. Thus, the decoder likelihood can equivalently be written as

$$p_{\theta_d}(\mathbf{x}_t | \mathbf{z}_t) \equiv p_{\hat{\phi}}(\mathbf{x}_t | \mathbf{z}_t),$$

emphasizing that the dependence on θ_d enters through the decoder-generated distributional parameters $\hat{\phi}$.

Both the encoder and decoder are multilayer perceptron (MLP) neural networks whose weights and biases are updated iteratively to maximize the differentiable Evidence Lower Bound (ELBO) which is given by

$$\mathcal{L}(\theta_e, \theta_d) = \sum_{t=1}^T \mathbb{E}_{q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t)} [\log p_{\theta_d}(\mathbf{x}_t | \mathbf{z}_t)] - \sum_{t=1}^T D_{\text{KL}} [q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t) \| p_{\theta_d}(\mathbf{z}_t)], \quad (46.6)$$

where $p_{\theta_d}(\mathbf{z}_t)$ is the prior on $\mathbf{z}_t$:

$$p_{\theta_d}(\mathbf{z}_t) \equiv p_{\hat{\phi}}(\mathbf{z}_t) = \prod_{k=1}^K h(z_{kt}; \hat{\alpha}, \hat{\gamma}_k),$$

where the model parameters $\hat{\phi}$ are learned through the decoder network described in (46.5), which does not introduce additional latent variables; rather, it provides an amortized neural estimator of the dependence parameters. This corresponds to an empirical-Bayes estimation strategy for the max-ID model.

Equivalently, the ELBO can be written as

$$\mathcal{L}(\theta_e, \theta_d) = \sum_{t=1}^T \mathbb{E}_{q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t)} [\log p_{\theta_d}(\mathbf{x}_t, \mathbf{z}_t) - \log q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t)],$$

which is a variational lower bound on the marginal log-likelihood $\sum_{t=1}^T \log p_{\theta_d}(\mathbf{x}_t)$. The first term in (46.6) is the expected reconstruction log-likelihood and encourages the latent representation to retain information needed to reconstruct the data. The second term regularizes the approximate posterior so that it remains close to the prior distribution in the latent space.

Since the latent variables have distributions that enable a generative process, we can obtain many copies of the original fields $\{\mathbf{X}_t : t = 1, \dots, T\}$ that have similar properties, especially in the joint tail. Moreover, the decoder $p_{\hat{\phi}}(\cdot | \mathbf{z}_t)$, given a sampled latent variable vector $\mathbf{z}_t$ from the encoder, can utilize the basis function values

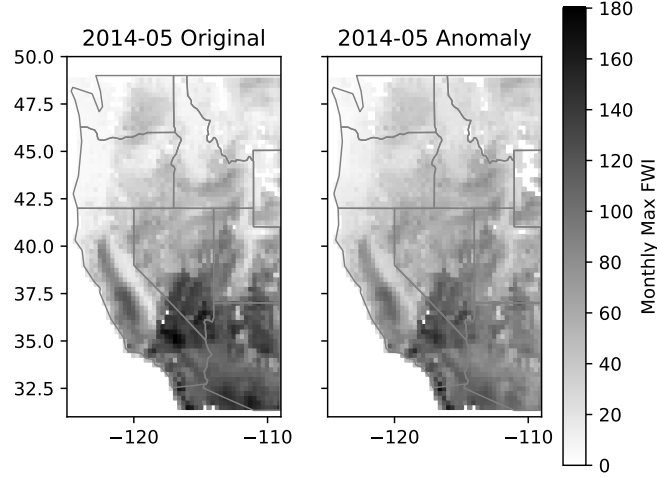


Figure 46.1: May 2014 monthly maxima fire weather index (FWI) values in the western US and its corresponding de-seasonalised anomalies

at the unobserved locations and therefore it naturally produces predictions at unobserved locations using the construction of the max-ID process. That is, $p_{\hat{\phi}}(\cdot | \mathbf{z}_t)$ is a predictive distribution at observed and unobserved locations, for each draw of $\mathbf{z}_t$. This ability to perform spatial predictions is crucial in producing spatially-complete data products and giving sensible risk assessments with proper uncertainty.

Lastly, we note that we use a heteroskedastic Gaussian variational family $q_{\theta_e}(\mathbf{z}_t | \mathbf{x}_t)$ for computational tractability, although the latent prior $p_{\theta_d}(\mathbf{z}_t)$ is specified through the expPS distribution. This Gaussian distribution should be interpreted only as an approximation to the posterior $p_{\theta_d}(\mathbf{z}_t | \mathbf{x}_t)$, rather than as a Gaussian replacement for the expPS prior. The heavy-tailed behavior of the model is retained through the expPS prior and the max-ID decoder. As explained in Appendix C.1 of [1851], this choice is also practically motivated: heteroskedastic Gaussian variational approximations are widely used in variational Bayes, and posterior distributions are often much more concentrated than their corresponding priors (due to the Bernstein–von Mises Theorem). Moreover, our compactly supported basis functions induce a localized dependence structure, which gives heuristic support for a Gaussian posterior approximation under spatial mixing-type arguments. In our experiments, heavier-tailed variational families were less stable and did not improve emulation performance, presumably because the expPS prior already provides the required tail flexibility, while an overly diffuse variational distribution can make the latent encoding less precise.

46.2.3 Application to the Fire Weather Index

The Fire Weather Index (FWI) is commonly used to estimate fire danger based on meteorological conditions, helping to anticipate and manage wildfire risks. A global FWI data product with a spatial resolution of $0.3125^\circ \times 0.25^\circ$ is available from NASA’s NCCS Data Portal at <https://portal.nccs.nasa.gov/datashare/GlobalFWI/>. Various remote sensing and *in situ* sources are integrated with fire weather variables such as temperature, wind speed, and humidity to produce the FWI as in [1690]. Note, conditions at some locations may not satisfy the assumptions required to compute the FWI and so some data are missing at each time. We consider monthly maxima at each location, which limits the amount of missing data (e.g., see Figure 46.1 for May 2014 monthly maxima in the 2,887 grid cells over the interior western U.S. region of interest here). For the analysis presented, we consider FWI monthly maxima for the period May 2014 through December 2024 (128 months), where any location with more than 50% NA records was discarded from subsequent analysis.

Data preprocessing

The extremes model used here assumes temporal independence and stationarity, and so seasonality was removed independently at each grid cell by fitting a 12-month cyclic cubic-spline generalized-additive model at each location and subtracting that from the monthly maxima at each FWI grid location to obtain FWI anomalies (see the right image in Figure 46.1 for an example of such anomalies).

In addition, for each anomaly series at a grid location ξ_i , we denote $\{\text{FWI}_{it}^a : t = 1, \dots, T\}$. If at least three

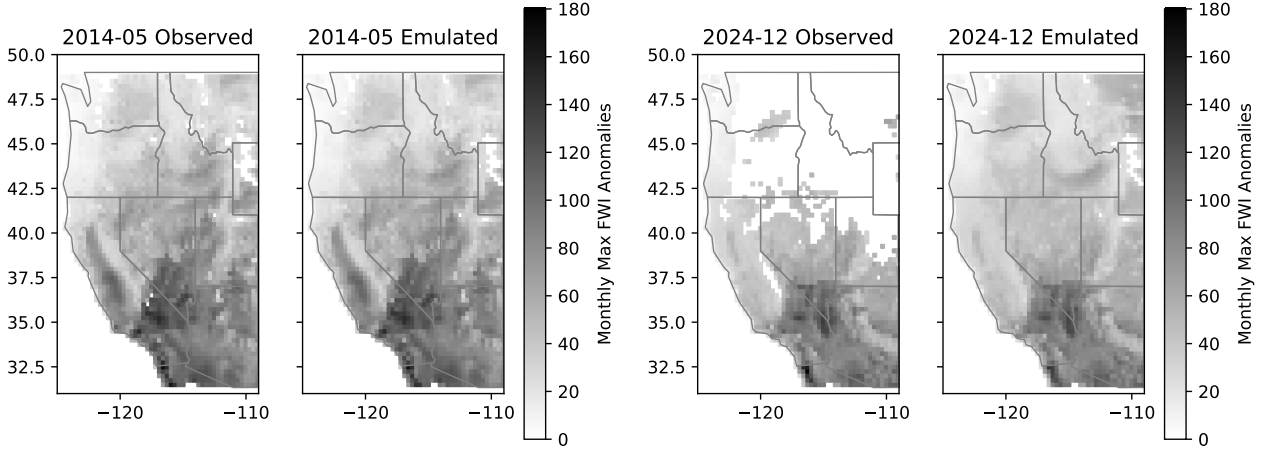


Figure 46.2: Monthly maxima FWI anomalies and corresponding emulated sample in May 2014 (left plots) and December 2024 (right plots).

non-missing values were available, a generalized extreme value (GEV) distribution was fit with the μ_i (location), τ_i (scale), and ζ_i (shape) parameters estimated by maximum likelihood and each anomaly series was monotonically transformed via

$$X_{it} = |\zeta_i(\beta_i - \text{FWI}_{it}^a)/\tau_i|^{1/\zeta_i}, t = 1, \dots, T, i = 1, \dots, m, \quad (46.7)$$

so that the X -data at each location is Pareto-tailed. Here, $\beta_i = \mu_i - \tau_i/\zeta_i$ is the upper bound of the marginal GEV distribution if $\zeta_i < 0$ and it is the lower bound if $\zeta_i > 0$. Therefore, the transformation (46.7) always defines a monotone transform. Once we obtain an emulated copy of $\mathbf{X}_t$ (i.e., a generated sample), we make an inverse transformation to get the emulation of $\{\text{FWI}_{it}^a : t = 1, \dots, T\}$.

FWI data emulation with xVAE

As described in Section 46.2.2, the xVAE uses a standard encoder-decoder architecture adapted to the max-ID spatial extremes model. The encoder takes the spatial field $\mathbf{x}_t = (x_t(\xi_1), \dots, x_t(\xi_{n_s}))^\top$ as input and maps it to the parameters of a diagonal Gaussian variational distribution for the latent vector $\mathbf{z}_t$. Specifically, the encoder is implemented as an MLP with nonlinear hidden layers,

$$\mathbf{h}_{1,t} = \text{ReLU}(\mathbf{W}_1 \mathbf{x}_t + \mathbf{b}_1), \quad \mathbf{h}_{2,t} = \text{ReLU}(\mathbf{W}_2 \mathbf{h}_{1,t} + \mathbf{b}_2),$$

followed by two output layers that return the posterior mean and log-variance,

$$\log \sigma_t^2 = \mathbf{W}_3 \mathbf{h}_{2,t} + \mathbf{b}_3, \quad \boldsymbol{\mu}_t = \text{ReLU}(\mathbf{W}_4 \mathbf{h}_{2,t} + \mathbf{b}_4).$$

A latent draw is then obtained through the reparameterization

$$\mathbf{z}_t = \boldsymbol{\mu}_t + \sigma_t \odot \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim N(\mathbf{0}, \mathbf{I}_K),$$

which enables stochastic-gradient optimization of the ELBO. The decoder maps the sampled latent vector $\mathbf{z}_t$ to the distributional parameters of the max-ID model, most importantly the dependence parameters $(\alpha, \boldsymbol{\gamma}^\top)^\top$. Thus, the encoder learns a low-dimensional latent representation of the spatial field, while the decoder translates this representation into interpretable extremal-dependence parameters used to reconstruct or emulate spatial extreme fields.

Figure 46.2 compares the observed and emulated monthly maximum FWI anomaly fields for May 2014 and December 2024. Grey scales are matched across panels for direct comparison. As introduced in Section 46.2.2, a key advantage of our approach lies in its ability to perform accurate spatial prediction, especially in the presence of missing data. Visual inspection shows strong agreement between the emulated and observed fields, with the model capturing both large-scale patterns and fine-scale variations. Note we deliberately left out locations in the December 2024 observations in Figure 46.2 that had over 50% NA records and were not included in the analysis.

To evaluate the quality of the emulations beyond visual assessment, we quantify tail dependence using explicit statistical measures, as discussed in the next section. These results indicate that the emulator captures both

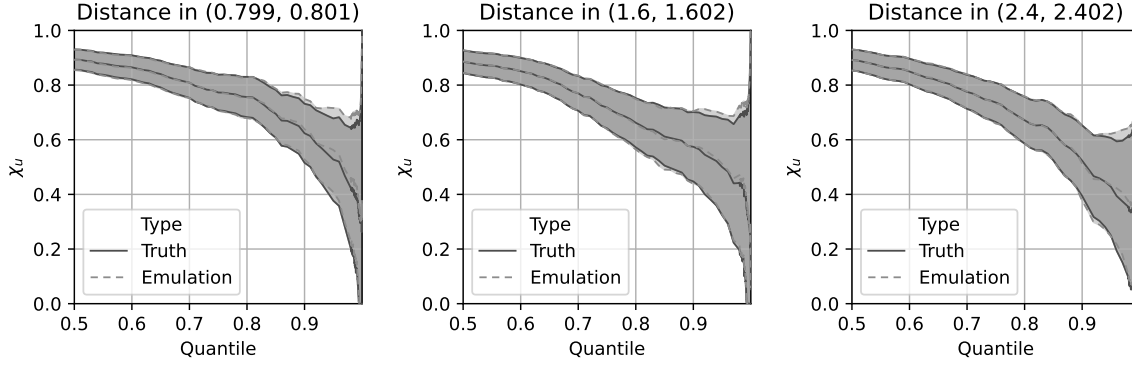


Figure 46.3: Comparison of the empirically-estimated $\chi_h(u)$ at $h = 0.8, 1.601, 2.401$ for 2015 FWI.

bulk variability and extremal dependence effectively. In addition, the framework developed in [1851] allows the dependence parameters $(\gamma_1, \dots, \gamma_K)^\top$ to vary over time. Temporal changes in these parameters, estimated via the decoder in Equation (46.5), can reveal trends in extremal dependence structure.

46.2.4 Tail-Dependence Validation with Extremal Dependence Measures

In the spatial extremes literature, the extremal dependence between two random variables X_1 and X_2 is commonly quantified using the χ -measure, defined as the conditional exceedance probability:

$$\chi_{12}(u) = \Pr\{F_2(X_2) > u \mid F_1(X_1) > u\} = \frac{\Pr\{F_2(X_2) > u, F_1(X_1) > u\}}{\Pr\{F_1(X_1) > u\}}, \quad (46.8)$$

where $u \in (0, 1)$, and F_1, F_2 denote the marginal distribution functions of X_1 and X_2 , respectively. This measure captures the likelihood of an extreme event at one location occurring given an extreme event at another. If $\chi_{12} := \lim_{u \rightarrow 1} \chi_{12}(u) = 0$, the pair is said to be *asymptotically independent*; if $\chi_{12} > 0$, they are *asymptotically dependent*. This distinction is critical in assessing joint risks of spatially dependent extremes such as floods or heatwaves.

To evaluate whether the xVAE realistically reproduces spatial upper-tail dependence, we assume that $\{X_t(\xi)\}$ exhibits a stationary and isotropic dependence structure at each time point t , so that $\chi_{12}(u) \equiv \chi_h(u)$, where $h = \|\xi_1 - \xi_2\|$ is the distance between locations. For a given lag h , we identify all location pairs whose distances fall within a small tolerance (e.g., $\epsilon = 0.001$), and compute empirical estimates of $\chi_h(u)$ over a grid of high quantiles u , pooling across all 12 monthly replicates. This surely violates our model assumption (our max-ID model is non-stationary over space), but the resulting diagnostic is still informative because it provides a low-dimensional, domain-averaged summary of how upper-tail dependence decays with distance and changes across quantile levels. Since the same summary is computed for both the observed fields and the xVAE emulations, agreement between the two indicates that the emulator preserves the main distance-dependent features of extremal dependence, even if the underlying dependence structure is spatially heterogeneous.

Figure 46.3 presents the estimated $\chi_h(u)$ curves for both the observed data and xVAE emulations across three spatial lags $h = 0.8, 1.601, 2.401$ for 2015 (with tolerance $\epsilon = 0.01$); other years show similar results. Grey-shaded bands represent 95% confidence intervals for the observation-based estimates. Across all cases, the xVAE reproduces the extremal dependence structure with remarkable fidelity, underscoring its capability to preserve tail dependence across space.

Despite its importance, tail dependence validation is rarely addressed in the emulation literature. While many proper scoring rules exist for assessing predictive performance [639], these typically average over all quantiles and may miss deficiencies in the tail. As highlighted in [1851], specialized rule-based metrics can more effectively assess both bulk and tail dependence. Accurate characterisation of tail behaviour is essential for credible risk assessment and for capturing the spatial footprint of extreme events.

46.2.5 Posterior Inference: Areal Exceedance Probabilities

As briefly commented in Section 46.2.2, one advantage of xVAE is that it can perform spatial predictions and generate a huge number of replicates from the approximate posterior distribution rapidly. Therefore, we can use

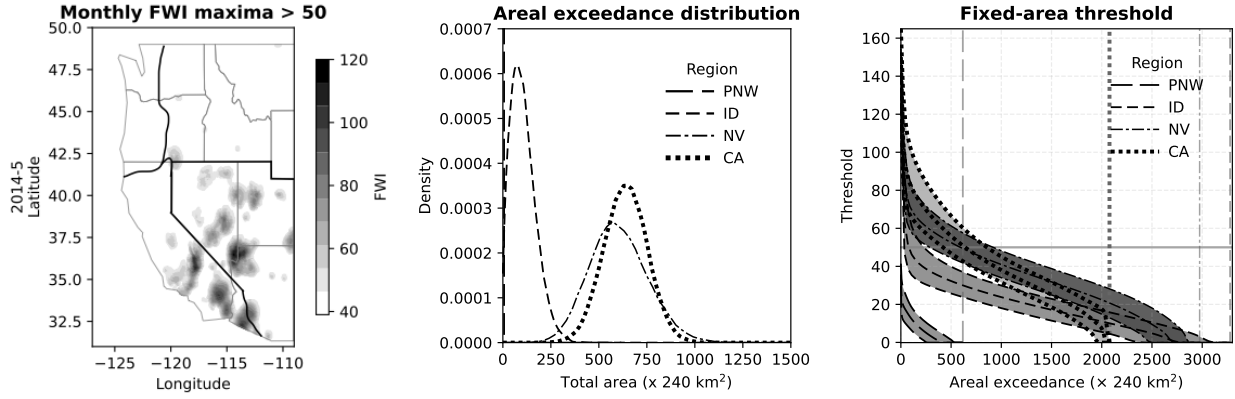


Figure 46.4: The left panels show realisations of monthly FWI maxima emulated with fitted parameters from the xVAE for May 2014. The emulations are censored with a threshold of 50. From 30,000 such emulations, we estimate the distribution of total area exceeding 50 within each region. The western U.S. is divided into four regions according to the ecological regions level I division [1777]: PNW, Idaho region (ID), CA, Nevada region (NV). On the right, we estimate the threshold it takes to have a fixed area of exceedance. The 95% approximate credible intervals are also shown. The vertical dashed lines are total areas of each subregion, and the horizontal slices at 50 yield results that align with the middle panel.

these simulated copies to calculate an important metric for dependence extremes—areal exceedance probabilities—which represents the spatial extent of a region simultaneously at extreme wildfire risk. Using the trained xVAE, we generate 30,000 independent monthly maximum FWI for each time point and compute the total area exceeding 50, which corresponds to extreme fire danger according to operational standards in the Canadian Forest Fire Weather Index system [1565, 1690]. The left panel of Figure 46.4 shows one of the monthly maximum FWI realisations emulated from the xVAE for May 2014 thresholded at $\text{FWI} > 50$. The middle panel shows the probability density functions of total area where FWI exceeds 50 (in units of 240 km^2) within each of four ecological subregions: Pacific Northwest (PNW), Idaho/Montana (ID), California (CA), and Nevada/Arizona/New Mexico (NV), based on the 30,000 sampled fields.

The results reveal substantial regional heterogeneity in the extent of FWI exceedances, with California showing the most persistent large-area exceedances, while the PNW and ID exhibit much lower exceedance volumes. For PNW in particular, the fire risk is consistently low and there is zero areal exceedance of the threshold 50. While the ID subregion has the largest land mass, its areal density concentrates around 70 ($\times 240 \text{ km}^2$), much lower than the CA and NV subregions.

To further analyze joint exceedances at varying extreme levels, the rightmost panels of Figure 46.4 invert the typical fixed-threshold paradigm by asking: *What FWI level corresponds to a given exceedance area?* That is, for each fixed spatial extent, we calculate the minimal threshold needed to reach that area of joint exceedances from each emulated sample, and we then group all 30,000 estimated thresholds together to derive 95% approximate empirical credible intervals. We repeat this process for all spatial extents of exceedances in between 5 to 3,300 ($\times 240$) km^2 . Note that this can be computed rapidly thanks to the semi-amortized nature of our xVAE. This shows that within each subregion, the spatial extent of exceedances decreases as the threshold increases and extreme events becomes more localized as they get more extreme. Furthermore, as the spatial extent approaches zero, the threshold estimates represent the highest possible FWI for a specific month in a subregion, a valuable metric for wildfire planning.

These results underscore the ability of the xVAE to provide spatially-explicit, probabilistic risk assessments for wildfire danger by capturing both marginal and joint tail behaviour. By leveraging the high-resolution spatial modelling of the xVAE and its ability to generate massive posterior simulations, we obtain stable, data-informed estimates of both the likelihood and spatial scope of extreme fire weather events, which enable risk planners to adaptively set local thresholds based on desired levels of spatial risk coverage. The smooth shape and narrowing confidence bands of these curves highlight the model’s strength in providing flexible and robust uncertainty quantification. Together, these plots offer a rigorous and scalable framework for tailoring fire weather warnings and mitigation strategies across heterogeneous geographic regions and varying operational thresholds.

46.3 Application: Level-Set Dynamics with Bayesian Deep Echo State Network (ESN) for Modelling Fire Front Propagation

In this application, we illustrate a model for fire front propagation that is motivated by a mechanistic evolution equation for which a data-driven spatio-temporal spread rate term is critical. Importantly, we model this spread rate in terms of a deep echo state network (ESN). The advantage of the ESN is that only the output parameters need to be learned, which is necessary here because of our very small sample size (12 fire boundaries). Importantly, the model is implemented in a Bayesian framework to properly account for uncertainty, with a novel use of the so-called “cut-Bayes” approach [1331] for dealing with nuisance parameters. Such methods have not been applied to reservoir computing models to the best of our knowledge. We illustrate the model with the infamous Camp Fire, a 2018 devastating mega-fire in California. Before presenting the model and results, we give background on the level set approach and the ESN.

46.3.1 Level Set Background

The level set approach consists of a partial differential equation that describes the evolution of an implicit function in space defined as:

$$\frac{\partial \psi(\xi, t)}{\partial t} + v(\xi, t) \cdot \nabla \psi(\xi, t) = 0, \quad (46.9)$$

where $\psi(\xi, t)$ denotes an implicit function at spatial location $\xi \in \mathbb{R}^d$ and time $t \in \mathbb{R}$, and $\nabla \psi = (\partial \psi / \partial \xi_1, \dots, \partial \psi / \partial \xi_d)^\top$ is the gradient of ψ over space. The evolution of the process $\psi(\xi, t)$ in (46.9) is deterministic and primarily driven by the spatio-temporal varying velocity field $v(\xi, t) \in \mathbb{R}^d$. Modelling this velocity requires care in order to accurately capture the evolution of the underlying implicit function. In general, there is no closed form solution to Equation (46.9) and numerical methods must be used for its solution [1556].

It is typically a realistic assumption in wildfire modelling that the fire evolves in the direction normal to the boundary [20, 1126, 1818, 1819], in which case Equation (46.9) simplifies to:

$$\frac{\partial \psi(\xi, t)}{\partial t} + v_n(\xi, t) \cdot \|\nabla \psi(\xi, t)\| = 0, \quad (46.10)$$

where $v_n(\xi, t) \in \mathbb{R}$ denotes the scalar velocity (speed) in the *normal* direction and $\|\cdot\|$ denotes the Euclidean (L_2) norm. The implicit function of choice for $\psi(\xi, t)$ is typically a signed distance function [20, 1126, 1818, 1819]. That is, for $\xi \in \mathbb{R}^2$,

$$\psi(\xi) = \begin{cases} -d(\xi) & \text{if } \xi \in \Omega^-, \\ 0 & \text{if } \xi \in \partial\Omega, \\ d(\xi) & \text{if } \xi \in \Omega^+, \end{cases} \quad (46.11)$$

where a location ξ on the boundary $\partial\Omega$ has a signed distance of zero. The signed distance for points inside ($\partial\Omega^-$) and outside ($\partial\Omega^+$) the boundary is defined as the minimum distance from ξ to the boundary, with negative and positive signs, respectively. In wildfire modelling, $d(\xi)$ is defined by the minimum Euclidean distance

$$d(\xi) = \min(\|\xi - \xi_I\|), \quad (46.12)$$

where $\xi_I \in \partial\Omega$ is the set of points on the boundary. Importantly, the fire front is represented “implicitly” by the zero contour of the signed distance function, rather than by an explicit set of points on the fire front. This signed distance representation is simple to interpret and is very flexible in describing potentially complex boundaries (e.g., merges and splits). Essentially any boundary shape can be represented by the signed distance function and the details of the fire front evolution can be preserved in the signed distance function representation if one uses a relatively dense spatial grid.

Because the signed distance function has the property that $\|\nabla \psi(\xi, t)\| = 1$, we can simplify Equation (46.10) so that

$$\frac{\partial \psi(\xi, t)}{\partial t} + v_n(\xi, t) = 0, \quad (46.13)$$

which can be solved by a simple forward Euler numerical approximation:

$$\psi(\xi, t) = \psi(\xi, t - \Delta t) - v_n(\xi, t - \Delta t) \Delta t, \quad (46.14)$$

where Δt is the time discretisation interval. Equation (46.14) describes the process at time t in terms of its state at $t - \Delta t$ (Markov property) with the advection governed by the speed $v_n(\xi, t - \Delta t)$. The model for $v_n(\xi, t)$ is critical and should be flexible so as to capture realistic fire evolution behaviour. To achieve this, [1819] employed a latent Gaussian spatio-temporal dynamic process within a Bayesian hierarchical modelling framework, while [1818] leveraged the flexibility of a calibrated ESN to model speed. The ESN approach worked better in real-world wildfire examples, and so we consider it here, extending it to a deep ESN and incorporating this in a fully Bayesian framework.

46.3.2 ESN Background

Deep learning neural network models have improved prediction and classification tasks when data have complex relationships. As with many random effects-based statistical models, accounting for dependency in data is one of the important strengths of neural network models. In the case of sequence data (e.g., language, time series), recurrent neural networks (RNNs) provide a powerful modelling framework to capture complex serial dependence. Despite their flexibility, training such models via “backpropagation through time” [1743] proved to be challenging due to the vanishing/exploding gradient problem [449]. Practitioners have typically turned to gated long short-term memory (LSTM; [760]) models or attention mechanism [1669] as modern alternatives. An earlier alternative, the ESN [1094], has regained popularity in recent years due to its effectiveness in settings where data volume is insufficient for training LSTMs or attention-based deep models.

Put simply, the ESN is a special case of an RNN where the weights connecting the input to the hidden states and weights between hidden states are not learned, but are randomly initialized and remain fixed [1094]. Such models with random weights are considered a type of reservoir computing, with the randomly connected layers serving as the “reservoir.” An ESN with a sufficiently large number of hidden states has been shown to satisfy the universal approximation property, which means it can approximate any continuous function [711]. The most basic ESN for time $t = 1, \dots, T$ can be written as:

$$\begin{aligned}
 \text{output : } y_t &= W^{\text{out}} h_t, \\
 \text{hidden state : } h_t &= (1 - \alpha_{\text{leak}}) h_{t-1} + \alpha_{\ell} \tilde{h}_t, \\
 \tilde{h}_t &= g_h \left(\frac{\nu}{|\lambda_w|} W^{\text{in}} h_{t-1} + U x_t \right), \\
 \text{parameters : } W^{\text{in}} &= [w_{i,\ell}^{\text{in}}]_{i,\ell} : \gamma_{i,\ell}^w \text{Unif}(-a_w, a_w) + (1 - \gamma_{i,\ell}^w) \delta_0, \\
 U &= [u_{i,j}]_{i,j} : \gamma_{i,j}^u \text{Unif}(-a_u, a_u) + (1 - \gamma_{i,j}^u) \delta_0, \\
 \gamma_{i,\ell}^w &\sim \text{Bernoulli}(\pi_w), \\
 \gamma_{i,j}^u &\sim \text{Bernoulli}(\pi_u),
 \end{aligned} \tag{46.15}$$

where $y_t \in \mathbb{R}^N$, $h_t \in \mathbb{R}^J$, and $x_t \in \mathbb{R}^{n_u}$ denote the output, hidden states (reservoir), and inputs, respectively. In the reservoir framework, the sparse matrices $W^{\text{in}} \in \mathbb{R}^{J \times J}$ and $U \in \mathbb{R}^{J \times n_u}$ are selected randomly. For example, as shown in Equation (46.15), elements of these matrices can be drawn from a uniform distribution with range parameters a_w and a_u . In addition, the parameters π_w and π_u determine the sparsity of W^{in} and U , where δ_0 denotes the Dirac delta function at zero.

The ESN must satisfy the so-called “echo state property” [1839] to effectively model nonlinear processes. Intuitively, this property requires that h_t become insensitive to initial conditions over time, so that the network’s behavior is driven primarily by recent inputs rather than by its initial state. In practice, this condition is satisfied when the spectral radius of W^{in} is less than 1 [1094], which can be ensured by scaling W^{in} with a factor ν (with $0 < \nu < 1$), such that the resulting spectral radius is $\nu/\lambda_w < 1$, where λ_w is the largest eigenvalue of W^{in} . The element-wise activation function $g_h(\cdot)$ (e.g., tanh or sigmoid) introduces nonlinearity into the model, as in other neural network models. The recursive nature of the model allows the information in h_{t-1} to be propagated to h_t , and a leaking rate α_{leak} (where $0 \leq \alpha_{\text{leak}} \leq 1$) determines how much of the memory from h_{t-1} is retained in h_t .

An appealing aspect of ESNs relative to RNNs is their low computational cost and training stability, as only the readout matrix $W^{\text{out}} \in \mathbb{R}^{N \times J}$ needs to be learned, whereas RNNs require training all parameters, including weights and biases. By adding an ℓ_2 regularisation penalty to the mean squared error in the loss function, W^{out} can be estimated explicitly using the ridge regression formulation, without using gradient descent optimisation.

That is,

$$\begin{aligned}\hat{W}^{\text{out}} &= \arg \min_{W^{\text{out}}} \left[\frac{1}{N} \sum_{t=1}^T \{(y_t - W^{\text{out}} h_t)^\top (y_t - W^{\text{out}} h_t)\} + \rho \sum_{j=1}^J \|w_{\cdot j}^{\text{out}}\|^2 \right] \\ &= YH^\top (HH^\top + \rho I)^{-1},\end{aligned}\tag{46.16}$$

where $Y = [y_1, \dots, y_T] \in \mathbb{R}^{N \times T}$ and $H = [h_1, \dots, h_T] \in \mathbb{R}^{J \times T}$ are matrices that concatenate y_t and h_t as columns for $t = 1, \dots, T$, respectively. Also, $w_{\cdot j}^{\text{out}}$ denotes the j^{th} column of W^{out} , $\|\cdot\|$ is the ℓ_2 norm, and the regularisation parameter $\rho \in (0, \infty)$ controls the regularisation shrinkage to 0.

Hyperparameter selection

The predictive performance of an ESN depends on its hyperparameters, as is typical for machine learning methods. [1094] recommend assigning small values to π_w and π_u and using a large reservoir size J to promote sparsity in W^{in} and U . The additional model hyperparameters, such as α_{leak} , ν , and ρ , can be optimized using validation or cross-validation techniques with a grid search [1158], genetic algorithms [1159], a backtracking search algorithm [1716], Bayesian optimisation [75, 1101, 1394], or a recycle validation approach [1358]. Alternatively, instead of optimizing hyperparameters, [1818] and [670] sampled them from their prior distributions. A review of ESN hyperparameter optimisation is given in [1150].

Uncertainty quantification

Uncertainty quantification for deep neural models has been an increasingly important research topic in recent years. For ESNs, a simple approach is the “ensemble method”, where uncertainty is evaluated using ensembles of forecasts generated from the reservoir given different reservoir weights [1158, 1511, 1810]. That is, each forecast is associated with W^{in} and U , which are independently drawn from their prior distributions given in Equation (46.15). The mean or quantiles of the ensembles can then be obtained from the collection of forecasts. Such methods can also include more sophisticated calibration (e.g., [184]). Alternatively, Bayesian methods can be used, e.g., [262, 1159].

ESN extensions

The ESN presented in Equation (46.15) is very basic. More advanced ESNs have been implemented in the literature for specific applications. For example, [1158] included a quadratic term h_t^2 and the corresponding readout matrix in the output layer to account for quadratic interactions in many physical and biological processes. Deep ESNs have also been used to accommodate multiple scales in the dynamics [185, 890, 1159]. A deep ESN consists of L hidden layers, where the output of the $(\ell - 1)^{\text{th}}$ hidden layer is fed into the ℓ^{th} hidden layer as input. This allows the model to capture more complex temporal dynamic patterns (we utilize a deep ESN in our model presented in Section 46.3.3). Furthermore, as physics-informed neural network models have provided a new paradigm for incorporating mechanistic relationships into neural networks [1371], the same ideas can be applied to ESNs. For example, the physics-informed ESN in [433] accomplishes this by regularizing the readout matrix to ensure that the predictions remain consistent with a specified mechanistic relationship. A systematic review of ESN design is provided in [1573]. Finally, ESNs can be used for other types of data support such as areal-level spatial data [1715] and count observations [1709]. Our approach given below provides a different approach for combining mechanistic information, deep ESNs, and a Bayesian inferential framework.

46.3.3 Bayesian Hybrid Level Set/Deep ESN Model

We now combine the discretised level set evolution equation for the signed-distance function representation of a fire front with a deep ESN to model the rate of spread normal to the fire front. The model is fully Bayesian and the random weights are sampled (although not learned) using a “cut-Bayes” procedure [1331]. To our knowledge, this is the first use of this approach for reservoir computing.

As indicated in Section 46.3.1, modelling the rate of spread $v_n(\xi, t)$ across space and time is crucial, as it determines the propagation of the fire front. To this end, we employ a deep ESN to infer $v_n(\xi, t)$ in Equation

(46.13). The full deep modelling structure is given as:

$$\begin{aligned}
\text{Response : } \psi_t &= \psi_{t-\Delta t} - v_{n,t-\Delta t} \Delta t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I), \\
\text{Rate of spread : } v_{n,t-\Delta t} &= \Psi W^{\text{out}} h_{t-\Delta t}^{(L)}, \\
\ell^{\text{th}} \text{ Hidden state : } h_{t-\Delta t}^{(\ell)} &= (1 - \alpha_{\text{leak}}^{(\ell)}) h_{t-2\Delta t}^{(\ell)} + \alpha_{\text{leak}}^{(\ell)} \tilde{h}_{t-\Delta t}^{(\ell)}, \\
&\tilde{h}_{t-\Delta t}^{(\ell)} = g_h \left(\frac{\nu^{(\ell)}}{|\lambda_w^{(\ell)}|} W^{\text{in},(\ell)} h_{t-2\Delta t}^{(\ell)} + U^{(\ell)} h_{t-\Delta t}^{(\ell-1)} \right), \\
1^{\text{st}} \text{ Hidden state : } h_{t-\Delta t}^{(1)} &= (1 - \alpha_{\text{leak}}^{(1)}) h_{t-2\Delta t}^{(1)} + \alpha_{\text{leak}}^{(1)} \tilde{h}_{t-\Delta t}^{(1)}, \\
&\tilde{h}_{t-\Delta t}^{(1)} = g_h \left(\frac{\nu^{(1)}}{|\lambda_w^{(1)}|} W^{\text{in},(1)} h_{t-2\Delta t}^{(1)} + U^{(1)} x_{t-\Delta t} \right), \\
\text{Parameters : } W^{\text{in},(\ell)} &= [w_{i,j}^{\text{in},(\ell)}], \quad w_{i,j}^{\text{in},(\ell)} \sim \gamma_{i,j}^{w,(\ell)} \text{Unif}(-a_w^{(\ell)}, a_w^{(\ell)}) + (1 - \gamma_{i,j}^{w,(\ell)}) \delta_0, \\
U^{(\ell)} &= [u_{i,j}^{(\ell)}], \quad u_{i,j}^{(\ell)} \sim \gamma_{i,j}^{u,(\ell)} \text{Unif}(-a_u^{(\ell)}, a_u^{(\ell)}) + (1 - \gamma_{i,j}^{u,(\ell)}) \delta_0, \\
\gamma_{i,j}^{w,(\ell)} &\sim \text{Bernoulli}(\pi_w^{(\ell)}), \\
\gamma_{i,j}^{u,(\ell)} &\sim \text{Bernoulli}(\pi_u^{(\ell)}),
\end{aligned} \tag{46.17}$$

where $\psi_t = (\psi(\xi_1, t), \dots, \psi(\xi_N, t))^\top$ represents the observed signed distance function for $\xi_1, \dots, \xi_N$ at time t and the rate of spread in the normal direction to the boundary at ξ and $t - \Delta t$ is given by $v_{n,t-\Delta t} = (v_n(\xi_1, t - \Delta t), \dots, v_n(\xi_N, t - \Delta t))^\top$. Additionally, $\epsilon_t = (\epsilon(\xi_1, t), \dots, \epsilon(\xi_N, t))^\top$ denotes an error term, and recall Δt is the time step size in the forward Euler discretisation. The rate of spread $v_{n,t-\Delta t}$ is parameterized by projecting onto orthogonal global basis functions (e.g., empirical orthogonal functions, EOFs), which is given as the product of Ψ , the readout matrix W^{out} , and the hidden state $h_{t-\Delta t}$, where $h_{t-\Delta t}$ is generated based on the structure of a deep ESN. Other parameters in the hidden state are defined in Section 46.3.2.

As with the ESN regularized by a ridge penalty, a closed-form solution can be derived as shown in (46.16). Uncertainty can be quantified using one of several *ad hoc* approaches as described in Section 46.3.2. Although such approaches have been quite effective, they lack the theory-based approach for uncertainty quantification that a fully Bayesian model provides.

Bayesian implementation

For the hierarchical model (46.17), a Bayesian implementation seems straightforward on first glance. Namely, the posterior distribution for W^{out} , σ^2 , and the other parameters $\theta_\ell = (\alpha_{\text{leak}}^{(\ell)}, \nu^{(\ell)}, a_u^{(\ell)}, a_w^{(\ell)}, \pi_w^{(\ell)}, \pi_u^{(\ell)})^\top$ for $\ell = 1, \dots, L$ can be written as $\pi(W^{\text{out}}, \sigma^2, \theta_{1:L} | \psi_{1:T})$, where $\psi_{1:T} = (\psi_1, \dots, \psi_T)$ and $\theta_{1:L} = (\theta_1, \dots, \theta_L)$. However, updating parameters using standard MCMC methods (e.g., Metropolis-Hastings) or Hamiltonian Monte Carlo suffer from identifiability issues and critically, we are not interested in updating the ESN input or reservoir weights and associated hyperparameters because their role is only to increase the ESN's expressiveness - they are nuisance parameters. One alternative is to simply sample these weights at each iteration of a sampler, accounting for their variability. But this does not produce a valid posterior distribution without further modification, which is known as the "cut-Bayes" approach [1331]. Intuitively, this approach "cuts" the dependence between parameters and data, so that data is not involved in updating the associated parameters. In our case, the "cut-Bayesian posterior" is given by:

$$\pi^{\text{cut}}(W^{\text{out}}, \sigma^2, \theta_{1:L} | \psi_{1:T}) \propto \pi(W^{\text{out}}, \sigma^2 | \psi_{1:T}) \pi^{\text{cut}}(\theta_{1:L}), \tag{46.18}$$

where $\pi^{\text{cut}}(\theta_{1:L})$ denotes the cut distribution for $\theta_{1:L}$, which is simply the prior distribution in our implementation.

Using the multiple imputation strategy discussed in [1331], the posterior distribution can be approximated by repeatedly sampling $\theta_{1:L}$ from $\pi^{\text{cut}}(\theta_{1:L})$ and subsequently updating σ^2 and W^{out} for each sampled value in a separate MCMC (Gibbs sampler). The posterior samples are then given by concatenating all samples from each

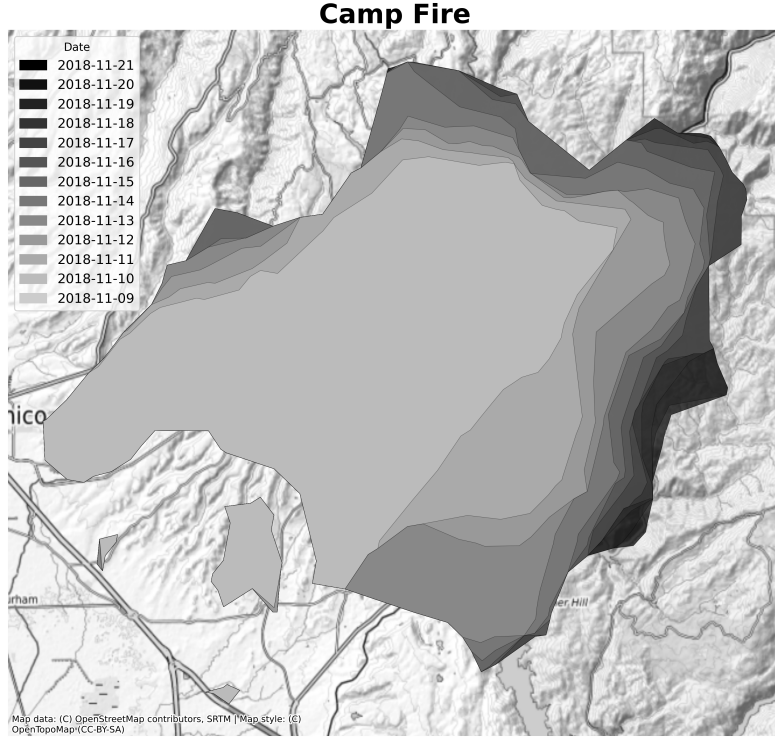


Figure 46.5: The evolution of the California (USA) Camp Fire from 9-20 November 2018.

of these MCMC runs. In this case, we assign prior distributions as:

$$\begin{aligned}
 W^{\text{out}} &\sim \mathcal{MN}(M_0, U_0, I), \\
 \sigma^2 &\sim \text{Inv.Gamma}(a, b), \\
 a_u^{(\ell)} &\sim \text{DU}\{0.1, 0.5, 1\}, \\
 a_w^{(\ell)} &\sim \text{DU}\{0.1, 0.5, 1\}, \\
 \alpha_{\text{leak}}^{(\ell)} &\sim \text{DU}\{0.01, 0.06, \dots, 0.94, 1\}, \\
 \nu^{(\ell)} &\sim \text{DU}\{0.1, 0.18, \dots, 0.81, 0.9\},
 \end{aligned} \tag{46.19}$$

where $\mathcal{MN}$ and DU denotes the matrix normal distribution and discrete uniform distribution, respectively. The multiple imputation approach is quite naive, and can be inefficient (given multiple MCMC samplers must be implemented), in which case other cut approaches can be used [1331]. However, in our case, since we are simply implementing a Bayesian regression conditioned on the cut parameters, it is quite efficient to sample from the associated Gibbs sampler for the size of problems of interest with wildfire front propagation.

46.3.4 Application to the 2018 California Camp Fire

In this section, we demonstrate the forecasting performance of our method using the 2018 Camp Fire in California as a case study. The 2018 Camp Fire was one of the most devastating fires in California history, resulting in 85 fatalities and the destruction of nearly 19,000 structures [212]. It was ignited on November 8 due to extremely dry fuels and strong winds and was fully contained on November 25, after burning approximately 153,336 acres. Figure 46.5 shows the evolution of the Camp Fire from November 9 to November 20, 2018. It illustrates the fire's rapid expansion from the initial ignition point, along with distinct “spot fires” appearing to the southwest of the main fire.

For the analysis, we obtain the signed distance field from the boundary data provided by [706]. Specifically, we compute the signed distance function on a 30×30 grid for longitude and latitude ranges $[-121.7844, -121.3351] \times [39.5915, 39.9068]$ for each day from November 9 to November 20, 2018 (i.e., $T = 12$). To assess one-step-ahead forecasting performance, we leave out the data corresponding to November 20, 2018 ($T = 12$).

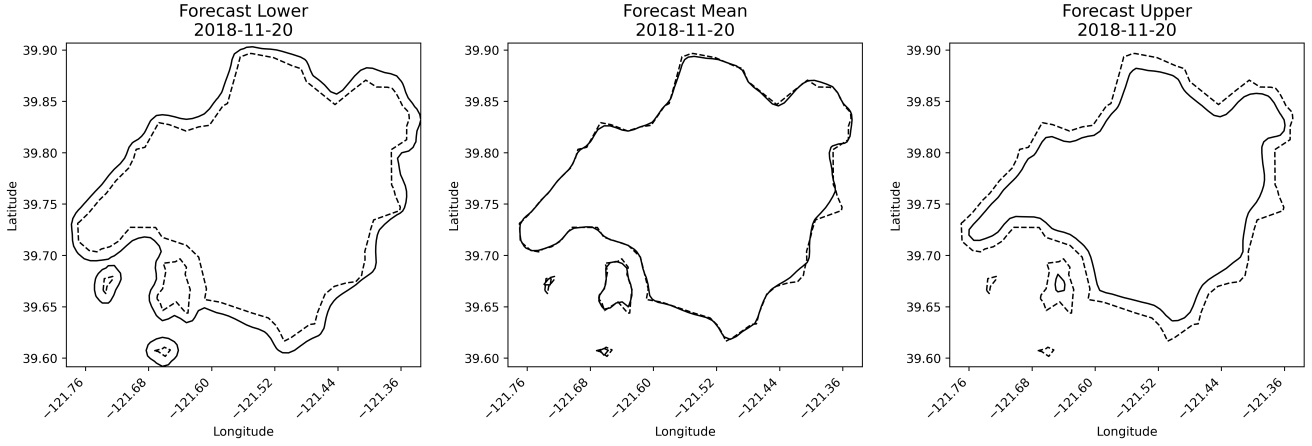


Figure 46.6: The posterior mean and 95% credible interval contours for the one-step-ahead out-of-sample forecast for 20 November 2018. Dashed and solid lines represent the true boundary and the forecast, respectively.

In the implementation, we use two hidden layers in the deep ESN, each with $J = 40$ hidden units. The prior distributions specified in (46.19) are used with $\tilde{M}_0 = \tilde{0}$, $U_0 = 10I$, and $a = b = 0.1$ where $\tilde{0}$ denotes a matrix of zeros. $\pi_w^{(\ell)}$ and $\pi_u^{(\ell)}$ are fixed at 0.3 and 0.5, respectively. We use the EOF basis functions [345] for Ψ calculated from the in-sample values of ψ_t , truncated to the first 5 modes. For the input, we use lagged values of the data, $x_{t-\Delta t} = (\psi_{t-\Delta t}^\top, \psi_{t-2\Delta t}^\top)^\top$, following [1158]. We define $h_1^{(1)}$ and $h_1^{(2)}$ to be $U^{(1)}x_{1+\Delta t}$ and $U^{(2)}h_1^{(1)}$, respectively. To approximate (46.18), we set the number of imputations to 100. For each imputed set of parameters, we draw 5,000 posterior samples of W^{out} and σ^2 using Gibbs sampling, discarding the first 3,000 as burn-in.

Figure 46.6 shows the posterior mean and 95% credible intervals for the one-step-ahead forecasts. The posterior mean closely follows the true boundary, and the 95% credible intervals capture the “spot fires” in the lower left portion of the domain. Forecasting performance is evaluated in terms of Threat Score (TS) [1749] and Interval Score (IS) [639]. The TS score ranges from 0 to 1, with 1 indicating a perfect forecast. Since our primary interest is the fire front, we consider an event to have occurred if $\psi(\xi, t) \leq 0$. We compute the point-wise IS only at locations within the true fire front and then average them over space, noting that a smaller interval score is better. According to both summary measures, our model demonstrates better forecasting performance compared to the (non-deep neural) dynamic linear model Bayesian implementation of [1819] applied to these data. Specifically, the model presented here gives $TS = 0.9702$ and $IS = 0.0219$, compared to $TS = 0.9691$ and $IS = 0.0782$ for the model in [1819]. Note that the key difference between the two models lies in how the rate of spread is modelled. While [1819] model the rate of spread using spatially varying covariates and a first-order vector autoregression structure to capture linear relationships, the hybrid deep ESN model accounts for nonlinearity through activation functions in the hidden states. These results indicate that modelling nonlinear relationships plays an important role in accurately forecasting fire behaviour.

46.4 Conclusion

Wildfires are increasing in frequency and intensity worldwide, given dramatic climate and land use changes. There are several critical modelling tasks associated with mitigating the risks of wildfires. Here, we consider the problem of performing inference on wildfire risks associated with the fire weather index, implementing a novel flexible model for spatially dependent extremes in a variational Bayesian context, using a novel extreme-aware variational autoencoder (xVAE). We also demonstrate a model that can forecast fire front evolution using a combination of level set dynamics and a deep ESN to model the speed in the normal direction to the fire front. This model is implemented in a fully Bayesian context, with the nuisance ESN reservoir parameters accounted for via a cut Bayes approach. In both cases, we demonstrate realistic uncertainty quantification and efficient implementation through hybrid statistical/deep neural spatio-temporal models.

Several limitations and future research directions remain. First, the xVAE described in this chapter is primarily designed as an emulator of spatial extreme fields learned from the available training data. Although the decoder can generate fast high-dimensional replicates after training, the current formulation does not explicitly condition the latent distribution or decoder parameters on external covariates, such as seasonality, large-scale climate indices,

radiative forcing, or other physical state variables. This limits its direct use for scenario-specific simulation or extrapolation under changing environmental conditions. A promising extension is a conditional xVAE, in which the encoder, latent prior, and decoder are all allowed to depend on a covariate vector $\mathbf{c}_t$.

Second, the algorithm presented for the Bayesian hybrid level set/deep ESN model can become computationally expensive for large T . This is not suitable for operational wildfire management, highlighting the need for scalable algorithms. This need has also been emphasized by [1820], who proposed an online learning algorithm for the hybrid level set/ESN model for real-time forecasting. However, their approach was developed within a non-Bayesian framework, and the uncertainty quantification was somewhat *ad hoc*. Therefore, extending the proposed Bayesian hybrid level set/deep ESN model to an online learning framework, for example based on Sequential Monte Carlo, remains an important direction for future research.

Chapter 47

Software for BDL

This chapter outlines the software foundations that made Bayesian deep learning (BDL) scalable. Probabilistic programming, automatic differentiation, and accelerators now enable efficient posterior inference in neural networks. Frameworks like TensorFlow Probability, Pyro, NumPyro, and PyMC unify model specification, inference, and diagnostics, integrating classical Bayesian methods with deep-learning workflows. These tools implement scalable algorithms—variational inference, MCMC, the Laplace approximation, and ensemble methods—on modern hardware, making Bayesian reasoning a practical, reproducible part of machine learning systems.

47.1 Introduction

Bayesian deep learning (BDL) unites the flexibility of neural networks with the reasoning of Bayesian inference. By placing probability distributions over parameters, latent representations, or outputs, BDL provides not only predictions but also a measure of confidence in those predictions. This uncertainty-aware perspective is essential in domains such as autonomous systems, medicine, and scientific discovery, where understanding what a model does not know can be as valuable as its point estimates.

Early Bayesian neural networks [1108] demonstrated the appeal of marginalising over weights rather than optimising them, but computational limits made such methods impractical for decades. The situation changed with modern automatic differentiation, GPU acceleration, and probabilistic programming. Frameworks now combine expressive architectures with scalable posterior inference, allowing stochastic variational inference (SVI), Hamiltonian Monte Carlo (HMC), or Laplace approximations to run efficiently on contemporary hardware. These advances made Bayesian inference not just a theoretical concept but a practical and widely applicable approach for deep learning.

Framework	Backend	Inference Methods	GPU / TPU Support	Modeling Style
TensorFlow Probability	TensorFlow / Keras	VI, HMC, NUTS, SMC	(GPU, TPU)	Hybrid: declarative + functional
Pyro	PyTorch	SVI, HMC, NUTS	(GPU)	Imperative (stochastic functions)
NumPyro	JAX / XLA	SVI, HMC, NUTS, SMC	(GPU, TPU, multi-chain)	Functional, JIT-compiled
PyMC	PyTensor / JAX	VI (ADVI), HMC, NUTS	(GPU via JAX)	Declarative (model objects)
Stan	C++ / CmdStanPy	HMC, NUTS, VB	Partial (OpenCL)	Declarative (static model blocks)
Turing.jl	Julia / Flux	HMC, NUTS, SMC, VI	(GPU)	Hybrid (DSL + functional)

Table 47.1: **Probabilistic frameworks for BDL.** Overview of major ecosystems, summarizing their computational backends, inference methods, accelerator support, and modeling paradigms.

Software plays a central role in this transformation. Probabilistic programming languages (PPLs) such as TensorFlow Probability, Pyro, NumPyro, PyMC, and Stan provide unified abstractions for model specification, inference, and validation. Their design philosophy differs from that of classical machine-learning libraries: rather than defining deterministic computation graphs, they define generative processes with explicit randomness and automatic inference backends. In parallel, specialised toolkits for approximate Bayesian methods—Bayesian layers, variational posteriors, and Laplace approximations—extend existing deep-learning pipelines with uncertainty estimation at little additional cost. Recent frameworks further standardise ancillary processes such as estimating calibration metrics and evaluating out-of-distribution detection, as well as post-hoc calibration and conformal prediction, enabling the evaluation, benchmarking and deployment of probabilistic models at scale.

The landscape of Bayesian deep learning software is now broad and interconnected. General-purpose PPLs offer the flexibility to express complex hierarchical models, while dedicated libraries streamline practical uncertainty estimation for specific neural architectures and applications. This chapter surveys the most widely used frameworks that operationalise Bayesian deep learning. It emphasises software that is actively maintained, integrates with major deep-learning stacks, and supports scalable inference on modern accelerators. Our discussion covers probabilistic programming systems, approximate Bayesian methods, inference engines and diagnostics, and emerging application frameworks for uncertainty-aware optimisation and deployment.

47.2 Probabilistic programming ecosystems used for BDL

Before automatic differentiation (AD) [105] became a standard component of machine-learning frameworks, most probabilistic inference relied on symbolic gradients or manual derivations. This limited the complexity and scalability of probabilistic models. The integration of reverse-mode AD—first widely popularised through TensorFlow [1] and PyTorch [1297]—enabled gradient-based inference methods such as stochastic variational inference and Hamiltonian Monte Carlo to operate efficiently on large computational graphs. AD also made it feasible to define custom likelihoods, transformations, and reparameterisations without analytic derivatives, dramatically lowering the barrier between probabilistic and deep-learning ecosystems. Gradients are now the common foundation that both deep learning and probabilistic inference rely on. The probabilistic programming ecosystems that we present below, build upon the aforementioned automatic differentiation frameworks.

TensorFlow Probability. TensorFlow Probability (TFP) extends TensorFlow with native support for probabilistic modelling and inference. Its core *Distributions* and *Bijectors* libraries define probability densities, samplers, and composable transformations essential for normalising flows and reparameterisations [427]. TFP implements black-box variational inference, Hamiltonian Monte Carlo (HMC), and No-U-Turn Sampling (NUTS), and provides high-level `JointDistribution` abstractions for constructing complex models [1327]. For Bayesian neural networks, `tfp.layers` includes modules such as `DenseVariational` and `DenseFlipout`, which combine variational posteriors over weights with low-variance gradient estimators [1629, 1738]. Tight integration with Keras allows end-to-end training on GPUs and TPUs using standard optimisers, while tracking the evidence lower bound throughout optimisation.

```

1 import torch, pyro, pyro.distributions as dist
2 from pyro.infer import SVI, Trace_ELBO
3 from pyro.infer.autoguide import AutoDiagonalNormal
4 from pyro.optim import Adam
5
6 # 1) Toy regression data (X: [N, D], y: [N])
7 X = torch.randn(200, 5)
8 w_true, b_true = torch.randn(5), 0.5
9 y = X @ w_true + b_true + 0.1 * torch.randn(200)
10
11 # 2) Reuse standard DL components: nn.Linear becomes part of the Bayesian model
12 net = torch.nn.Linear(5, 1)
13 pyro.clear_param_store()
14
15 def model(X, y=None):
16     # 3) Register parameters from torch.nn with Pyro's param store
17     #     -> Pyro can learn variational posteriors over these weights
18     pyro.module("net", net)
19
20     # 4) Prior on observation noise (log-normal ensures positivity)
21     sigma = pyro.sample("sigma", dist.LogNormal(0., 0.1))
22 
```

```

23     # 5) Likelihood: predictions from net; to_event(1) treats batch as one event
24     mean = net(X).squeeze(-1)
25     pyro.sample("obs", dist.Normal(mean, sigma).to_event(1), obs=y)
26
27 # 6) Automatic mean-field guide over all latent sites in 'model'
28 guide = AutoDiagonalNormal(model)
29
30 # 7) SVI setup: Adam optimizer + ELBO objective
31 svi = SVI(model, guide, Adam({"lr": 1e-2}), loss=Trace_ELBO())
32
33 # 8) Optimization loop: each step updates variational params to minimize ELBO
34 for _ in range(500):
35     svi.step(X, y)

```

Listing 47.1: **Minimal Pyro example:** Bayesian linear regression trained with stochastic variational inference (SVI). The model reuses a `torch.nn` module inside a probabilistic model, illustrating Pyro’s seamless integration with deep-learning components. Key steps are annotated inline.

Pyro and NumPyro. Pyro [158] is a probabilistic programming language built on PyTorch. It combines flexible model design with scalable stochastic variational inference (SVI) and efficient MCMC methods such as NUTS and HMC. Through close integration with `torch.nn`, Pyro reuses existing architectures and training tools for Bayesian inference, making it broadly applicable and easy to use (see Listing 47.1 for a minimal example illustrating Pyro’s ease of use). NumPyro reimplements Pyro on JAX and XLA, providing the same abstractions with just-in-time (JIT) compilation and automatic vectorisation, making it ideal for large models and parallel sampling on accelerators [1323]. In both frameworks, automatic differentiation and batched linear algebra form the basis of scalable Bayesian learning.

BlackJAX. BlackJAX is a JAX-native [200] library for Bayesian inference built around a functional API that composes naturally with `jit`, `vmap`, and `pmap`. It implements HMC and NUTS as core kernels and includes additional Langevin, stochastic-gradient, sequential Monte Carlo, and variational methods within a unified interface. It also provides Microcanonical Langevin Monte Carlo (MCLMC) [1415], an automatically tuned sampler with fast burn-in that can outperform NUTS by one to two orders of magnitude in high dimensions and has been applied directly to BNN posteriors; it is currently available almost exclusively in BlackJAX. Operating directly on a user-defined unnormalised log density, BlackJAX integrates seamlessly with both probabilistic programming frameworks and hand-written Bayesian deep learning models in the JAX ecosystem. Its vectorised multi-chain execution on accelerators efficiently handles the high-dimensional posteriors common in BDL. See [224] for design details and algorithms.

PyMC. PyMC provides a Pythonic modelling interface for Bayesian models with implementations of state-of-the-art samplers such as NUTS and methods for variational inference such as ADVI [1138]. Although historically associated with hierarchical models, PyMC can also represent Bayesian neural networks by defining random variables for weights and biases, and it interoperates with the ArviZ [946] analysis stack for diagnostics and visualisation. PyMC excels for classical hierarchical models and small pedagogical BNNs, while frameworks such as `posteriors` [461] are more natural at practical deep-learning scale. For networks of moderate size, NUTS with mass-matrix adaptation often produces high-quality posteriors, while ADVI yields fast approximate solutions suitable for initialisation and model exploration.

Bean Machine and Stan. Bean Machine (now archived) provides a declarative modelling interface in the PyTorch ecosystem with automatic differentiation, variational, and Monte Carlo inference backends [1593]. Its design emphasises composability with deep-learning codebases. Stan offers a mature implementation of NUTS and extensive MCMC diagnostics [240]. Although Stan is less frequently used for high-dimensional BNNs due to memory and compile-time constraints, as well as limited support for custom likelihoods on accelerators, it remains a reference standard for smaller networks and methodological validation.

Turing.jl. Turing.jl is a general-purpose probabilistic programming language in Julia that provides flexible model specification and multiple inference backends. It supports gradient-based samplers and particle methods, and couples with `AdvancedHMC.jl` for modular implementations of HMC and NUTS with modern adaptation schemes [528, 1788]. By leveraging Julia’s automatic differentiation and interoperability with deep-learning libraries

Pyro (imperative / dynamic)

```

1 import pyro, pyro.distributions as dist
2
3 def model(X, y=None):
4     w = pyro.sample("w", dist.Normal(0., 1.).expand([X.size(1)]).to_event(1))
5     b = pyro.sample("b", dist.Normal(0., 1.))
6     sigma = pyro.sample("sigma", dist.HalfNormal(1.))
7     mean = X @ w + b
8     pyro.sample("y", dist.Normal(mean, sigma).to_event(1), obs=y)

```

TensorFlow Probability (declarative / static)

```

1 import tensorflow_probability as tfp
2 tfd = tfp.distributions
3
4 def model(X):
5     return tfd.JointDistributionNamed({
6         "w": tfd.Normal(0., 1.),
7         "b": tfd.Normal(0., 1.),
8         "sigma": tfd.HalfNormal(1.),
9         "y": lambda w,b,sigma: tfd.Normal(X @ w + b, sigma)
10    })

```

Figure 47.1: **Bayesian linear regression in two paradigms.** Pyro defines random variables imperatively within Python code; TFP declares them statically in a joint distribution structure.

such as Flux [816] and Lux [1284], Turing.jl suits Bayesian deep learning models by combining neural components with hierarchical structure. Users can mix HMC/NUTS for key subnetworks with likelihood-free or SMC kernels for non-differentiable parts within a single program, while still benefiting from GPU acceleration.

Probabilistic programming systems differ not only in syntax but also in modelling philosophy (see Table 47.1). Frameworks such as Pyro and NumPyro use an *imperative* or *functional* style, where stochastic variables are declared inline within Python or JAX code. This promotes flexibility and close integration with deep-learning workflows, supporting dynamic architectures and custom control flow. In contrast, systems like Stan and PyMC follow a more *declarative* approach: models are defined as static probabilistic graphs that can be analysed and optimised before inference. Declarative modelling typically yields more efficient inference and a clearer separation between model and algorithm, while imperative designs enable tighter coupling with neural components. TensorFlow Probability bridges the two by offering both graph-based and functional APIs, combining the scalability of static graphs with the flexibility of dynamic modelling. This division is not strict, however: functional JAX-based stacks such as BlackJAX occupy a middle ground, where explicit memory management and composability (`jit`, `vmap`, `pmap`) coexist with the freedom to mix hand-written log-densities and PPL-generated ones—an approach increasingly reflected in libraries like `posteriors` [461] that build on the functional PyTorch API.

47.3 Toolkits for approximate Bayesian deep learning

Bayesian layers and variational training. A pragmatic route to BNNs is to replace deterministic layers by Bayesian counterparts with mean-field or structured posteriors and train via variational objectives. TFP’s Bayesian layers implement these designs with reparameterisation and flipout estimators [1629, 1738]. Similar patterns are available in Pyro through auto-guides and custom guides that amortise posterior families over network weights [158]. These approaches prioritise speed and easy deployment, and they integrate well with existing model-selection and training pipelines.

Laplace approximations for neural networks. The Laplace approximation remains compelling for turning pretrained networks into approximate Bayesian predictors by fitting a local Gaussian around a mode. The `laplace-torch` library implements modern Laplace variants, including diagonal, Kronecker-factored, and approximations with scalable marginal-likelihood tuning [380]. This path yields calibrated uncertainty with minimal code changes and can be combined with ensembling or low-rank corrections.

Uncertainty estimation beyond probabilistic programming. A large portion of practical Bayesian deep learning research relies on approximate uncertainty estimation methods implemented outside traditional probabilistic programming frameworks. Techniques such as Monte Carlo dropout [566], deep ensembles [962], stochastic weight averaging [824] and its Bayesian variant SWAG [1116], as well as structured ensemble approaches like MaskEnsembles [469], BatchEnsemble [1739], and Packed-Ensembles [979], among many others, achieve competitive uncertainty quality and out-of-distribution robustness while remaining fully compatible with standard deep-learning pipelines. Each of these methods is accompanied by official implementations, and unified toolkits such as `uncertainty-baselines` [1220] and `torch-uncertainty` [959] further streamline evaluation and benchmarking across architectures and datasets. For tabular data, `bde` [51] implements Bayesian deep ensembles via MILE behind a familiar scikit-learn `fit/predict` API with a JAX backend.

Simulation-based inference. Simulation-based inference (SBI) methods [400] make it possible to learn inference algorithms requiring access only to model simulations, and therefore can also be applied to stochastic simulators with intractable likelihoods (Chapter 3). The `sbi` package implements a range of different inference approaches including neural posterior, likelihood, and ratio estimation, with diagnostics and samplers [178]; `BayesFlow` provides a framework focused on amortised Bayesian workflows [941]; and `swyft` targets simulation efficiency via truncated marginal ratio estimation [1176]. All provide methods which can amortise an upfront simulation and training cost into near-instant inference for new observations.

ArviZ. Diagnostic tools are crucial for BNNs, where non-identifiability and symmetries make convergence assessment challenging. ArviZ [946] is a Python library designed for the exploratory analysis of Bayesian models. It provides a unified, backend-agnostic interface for diagnosing, summarising, and visualising results from probabilistic programming frameworks such as PyMC, Stan, NumPyro, and Pyro. ArviZ includes tools for posterior predictive checks, convergence diagnostics, and rich statistical visualisations that help evaluate the quality and reliability of Bayesian inference. By standardising analysis workflows across different backends, ArviZ streamlines the interpretation and comparison of Bayesian model outputs. A caveat, however, is that many standard diagnostics are largely uninformative for BNN posteriors, whose weight-space symmetries and non-identifiability make them ill-suited to convergence assessment in the usual sense (see Chapter 33).

47.4 Applications and downstream toolchains

Bayesian optimisation. Beyond uncertainty estimation within neural networks, Bayesian principles also underpin optimisation and experimental design toolchains. `BoTorch` [78] provides a modular library for Bayesian optimisation built on PyTorch, supporting Gaussian process surrogates, neural network surrogates, and probabilistic acquisition functions. The example in Figure 47.2 illustrates a typical optimisation loop: the code listing (top) shows how a Gaussian process model and the Expected Improvement (EI) acquisition are used to propose new evaluation points, while the sequence of plots (bottom) visualises how the surrogate posterior progressively concentrates around the global optimum over successive iterations. `BoTorch` integrates seamlessly with `Ax` [1259], a high-level experimentation platform for automated hyperparameter tuning and adaptive data collection.

Active learning and dataset curation. Uncertainty estimates from Bayesian neural networks naturally enable active data acquisition strategies [1492]. By quantifying epistemic uncertainty, models can prioritise unlabelled or high-variance samples that most reduce posterior entropy. Libraries such as `modAL` [370] and `small-text` [1474] integrate BDL-style uncertainty measures—predictive entropy, mutual information, and ensemble prediction variance—into selection heuristics. When combined with scalable surrogate models or simulators, as in `BoTorch`, these methods support closed-loop data collection for vision, NLP, and scientific applications.

Out-of-distribution detection. Detecting when a model encounters inputs that differ from its training distribution is one of the most practical uses of uncertainty estimation. Bayesian deep learning methods approach out-of-distribution (OOD) detection by measuring predictive uncertainty—using posterior variance, entropy of the predictive distribution, or disagreement among ensemble members—as a signal of epistemic uncertainty. However, not all OOD detection pipelines require full Bayesian inference. The `PyOD` library [1866] provides a broad suite of classical and neural anomaly detection algorithms, including isolation forests, autoencoders, and deep density models, which can be combined with Bayesian uncertainty scores or calibrated posteriors.


```

1 import torch
2 from botorch.models import SingleTaskGP
3 from botorch.fit import fit_gpytorch_mll
4 from botorch.acquisition import ExpectedImprovement
5 from botorch.optim import optimize_acqf
6 from gpytorch.mlls import ExactMarginalLogLikelihood
7
8 # Objective function on [0,1]^2
9 def f(X):
10     return (torch.sin(6*X[:,0]) + torch.cos(6*X[:,1])).unsqueeze(-1)
11
12 # Initial design and GP fit
13 train_x = torch.rand(10, 2)
14 train_y = f(train_x)
15 gp = SingleTaskGP(train_x, train_y)
16 mll = ExactMarginalLogLikelihood(gp.likelihood, gp)
17 fit_gpytorch_mll(mll)
18
19 # Expected Improvement and candidate selection
20 EI = ExpectedImprovement(gp, best_f=train_y.max())
21 bounds = torch.stack([torch.zeros(2), torch.ones(2)])
22 candidate, acq_val = optimize_acqf(EI, bounds, q=1, num_restarts=5, raw_samples=128)

```

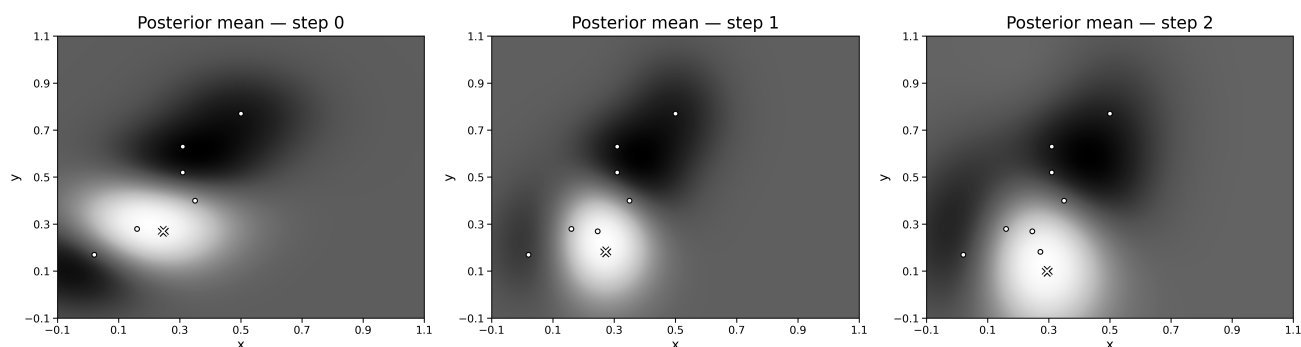


Figure 47.2: **Bayesian optimization of a 2D toy objective with a GP surrogate.** (Top): core BoTorch implementation showing model fitting and Expected Improvement optimization. (Bottom): posterior mean evolution across iterations — dots denote evaluated points, and the $\times$ marks the current best. The optimizer progressively refines its belief about the function’s optimum.

Bayesian neural architecture search and hyperparameter optimisation. Bayesian principles also govern the exploration of neural architectures and hyperparameters under uncertainty. Frameworks such as Ax [1259], Optuna [17], and Ray Tune [1039] employ probabilistic surrogate models—typically Gaussian processes or tree-structured Parzen estimators—to navigate large design spaces efficiently. Recent work integrates variational and Monte Carlo inference directly into architecture hyperparameters, treating network configurations as random variables with learned posteriors. These methods combine the flexibility of BDL with automated model selection, yielding uncertainty-based search procedures that adaptively balance exploration and exploitation across computational budgets.

Large language models and uncertainty estimation. The rise of large pretrained models has renewed interest in scalable uncertainty estimation. Bayesian fine-tuning techniques—such as Bayesian linear probes, low-rank variational adapters, and stochastic weight averaging—extend BDL concepts to foundation models while remaining computationally tractable. A rapidly growing line of work casts LoRA adapters as Bayesian, including Laplace-LoRA [1791] and SWAG-LoRA [1260]. In this context, the open-source framework LM_Polygraph [500] offers a unified toolkit for uncertainty estimation in text-generation tasks, supporting a variety of methods (e.g., token entropy, mutual information, sequence probability) and providing a benchmark for modelling and deployment (see Listing 47.2).

```

1 from transformers import AutoModelForCausalLM, AutoTokenizer
2 from lm_polygraph.utils.model import WhiteboxModel

```

```

3 from lm_polygraph.utils.manager import estimate_uncertainty
4
5 model_path = "bigscience/bloomz-560m"
6 tokenizer = AutoTokenizer.from_pretrained(model_path)
7 base_model = AutoModelForCausalLM.from_pretrained(model_path, device_map="cuda:0")
8 model = WhiteboxModel(base_model, tokenizer, model_path=model_path)
9
10 ue_method = "MeanPointwiseMutualInformation"
11 input_text = "What is the capital of France?"
12 ue = estimate_uncertainty(model, ue_method, input_text=input_text)
13 print(f"Generation: {ue.generation_text}")
14 print(f"Uncertainty score: {ue.uncertainty:.4f}")

```

Listing 47.2: LM_Polygraph for estimating uncertainty in large language models. This snippet loads a pretrained model (bloomz-560m), wraps it in the `WhiteboxModel` interface, and computes an uncertainty score using the `MeanPointwiseMutualInformation` method for a given input prompt.

47.5 Final remarks

Choosing a stack in practice. The choice depends on architectural preferences and target hardware. TFP and its Bayesian layers offer the most direct Keras integration, supporting end-to-end differentiable variational training and mature MCMC implementations. Pyro provides flexible model construction and scalable SVI around PyTorch modules, while NumPyro enables high-throughput NUTS and SVI on JAX with minimal modelling friction. For BDL specifically, the lower-level BlackJAX is often the better fit than the higher-level NumPyro, as it operates on arbitrary log-density functions and exposes cutting-edge kernels such as MCLMC that NumPyro does not wrap. PyMC offers a concise modelling interface with excellent diagnostics and is well suited for medium-scale BNNs and pedagogical settings. Laplace-based tools such as `laplace-torch` turn pretrained models into Bayesian predictors with minimal refactoring, and libraries like `Fortuna` [418] help standardise uncertainty evaluation and deployment. Since `Fortuna` is no longer actively maintained, we point readers to actively developed alternatives such as `torch-uncertainty` [959], `Lightning UQ Box` [1010], and `posteriors` [461] for this role.

Open challenges. Despite significant progress, Bayesian deep learning (BDL) still faces notable practical and conceptual obstacles:

- **Computational cost.** Inference algorithms such as Hamiltonian Monte Carlo (HMC) and full-rank variational inference (VI) remain computationally demanding for large neural architectures. Even with GPU batching and JIT compilation, runtime and memory consumption can be prohibitive for modern networks with tens of millions of parameters. Research on low-rank, structured, and subnetwork-based approximations continues to seek scalable compromises.
- **Reparameterisation and expressivity.** Reparameterisation gradients make VI efficient but restrict the space of admissible posterior families. Discrete latent variables and non-Gaussian priors often require specialised relaxations or reparameterisation tricks, which complicate implementations.
- **Evaluation metrics and calibration.** While predictive uncertainty is widely discussed, its quantitative evaluation remains inconsistent. Metrics such as negative log-likelihood, expected calibration error (ECE), Brier score, and coverage measure different aspects of uncertainty. Toolkits such as `Fortuna` and `Uncertainty Baselines` provide partial unification, yet widely accepted standardised benchmarks and evaluation protocols remain lacking.
- **Benchmark reproducibility.** Differences in initialisation, priors, datasets, and training schedules can drastically affect posterior quality and calibration. The community lacks a set of canonical, reproducible benchmarks for uncertainty estimation analogous to ImageNet or GLUE for accuracy-based evaluation. Shared evaluation suites and standardised reference implementations are essential for fair comparison and cumulative progress.
- **Storing and serving posterior samples.** Sampling-based BDL lacks widely adopted tooling for efficiently saving, compressing, sharing, and serving large collections of posterior weight samples. With millions of parameters and hundreds of samples, storage, I/O, and serving quickly become non-trivial, which partly explains why post-hoc and single-forward-pass methods still dominate in production.

Outlook. Ecosystems continue to converge on a small set of differentiable probability primitives, high-order transformations, and compiled samplers. For BDL, this unification means that modellers can prototype in a high-level PPL, borrow optimised inference kernels, and ship models with principled uncertainty estimates and reproducible diagnostics. The software landscape already supports most inference methods required in practice, and continued work on scalable samplers, structured variational families, and modular uncertainty evaluation will further lower the barrier to Bayesian methods in modern neural networks.

Future directions. Bayesian principles are increasingly converging with modern large-scale learning paradigms. The next generation of BDL software will likely be shaped by the following trends:

- **Foundation-model uncertainty.** Extending Bayesian fine-tuning to large language and vision models through variational low-rank adapters, Bayesian LoRA, and stochastic weight averaging across massive parameter spaces.
- **Scalable probabilistic inference.** Exploiting distributed JAX and PyTorch backends, multi-chain vectorisation, and compiler-level fusion to enable large-scale posterior sampling and amortised inference on accelerators.
- **Differentiable simulators.** Combining probabilistic inference with differentiable simulators to support Bayesian calibration of scientific models in physics, chemistry, and biology.
- **Uncertainty-aware deployment.** Extending probabilistic programming to production settings through standardised APIs for calibration, interpretability, and runtime uncertainty monitoring.

Together, these developments point toward a synthesis of classical Bayesian reasoning and modern foundation-model infrastructure, where uncertainty estimation becomes a native property of scalable machine-learning systems rather than a post-hoc addition.

Part X

Topical developments

Chapter 48

Subspace inference

Bayesian inference in modern neural networks is challenged by the high dimensionality of the parameter space Θ , which may contain millions of weights. General-purpose posterior inference in this space is often computationally demanding, while symmetries and redundancies can make posterior exploration difficult (see, e.g., Chapter 33). Subspace inference reduces this burden by restricting posterior uncertainty to a selected low-dimensional subset or parameterisation of Θ . Depending on the construction, this reduced space may be a linear subspace, a nonlinear manifold, or a more general low-dimensional region of the original parameter space.

48.1 Method overview

Various methods have been proposed for constructing such reduced parameterisations. In this chapter, we consider two approaches: *subnetwork inference* and inference in parameterisations constructed from connected modes of the loss surface.

Subnetwork inference uses an axis-aligned coordinate subspace by fixing some parameters and treating the remaining parameters as stochastic. Mode-connectivity approaches instead construct linear or nonlinear low-dimensional parameterisations that follow connected low-loss regions of the parameter space.

Once a reduced parameterisation has been selected, posterior inference can be performed over its coordinates using either sampling or an approximate inference method. A prominent example is last-layer Laplace inference, which treats the last-layer parameters as stochastic while fixing the feature extractor.

Related approaches retain the full parameterisation but design structured Markov chain Monte Carlo (MCMC) updates that reduce harmful posterior dependencies and improve mixing [21].

48.2 Subnetwork inference

A natural way to construct a reduced parameter space is to use the network’s topological structure as a guide. It has been well known for decades that neural networks can be substantially pruned, meaning that many of their weights can be set to zero, while incurring little or no loss in predictive performance [286]. This raises the question of whether uncertainty quantification in Bayesian neural networks (BNNs) follows a similar principle. In particular, can a Bayesian treatment be restricted to only a subset of the parameters while preserving the expressive benefits of a BNN?

In this section, we first summarise theoretical results showing that, under suitable assumptions, a Bayesian treatment can be restricted to a small subset of the parameters without sacrificing the expressivity of the resulting predictive distribution. We then describe a practical algorithm for selecting a subnetwork on which to perform Bayesian inference.

Let S and D denote disjoint index sets corresponding to the *stochastic* and *deterministic* parameters, respectively. A posterior distribution is placed over θ_S , whereas the parameters θ_D are fixed at values $c_D = (c_j)_{j \in D}$. Although we refer to the latter parameters as deterministic, their values may have been obtained using a learning algorithm or specified manually. The relevant distinction is that they are represented by point masses rather than non-degenerate posterior distributions. The resulting posterior can be written as

$$p(\theta_S, \theta_D \mid \mathcal{D}) = p(\theta_S \mid \mathcal{D}) \prod_{j \in D} \delta_{c_j}(\theta_j), \quad (48.1)$$

where δ_{c_j} denotes a point mass at c_j .

The corresponding posterior predictive distribution is

$$\begin{aligned} p(y_* | x_*, \mathcal{D}) &= \int_{\theta_S, \theta_D} p(y_* | x_*, \theta_S, \theta_D) p(\theta_S | \mathcal{D}) \prod_{j \in D} \delta_{c_j}(\theta_j) d\theta_S d\theta_D \\ &= \int_{\theta_S} p(y_* | x_*, \theta_S, c_D) p(\theta_S | \mathcal{D}) d\theta_S. \end{aligned} \quad (48.2)$$

The two central questions are therefore whether concentrating all posterior randomness in θ_S restricts the class of posterior predictive distributions, and how the stochastic subset S should be chosen in practice.

48.2.1 Expressivity of subnetwork-derived predictive distributions

One might expect subnetwork inference to trade predictive expressivity for computational tractability. However, Sharma et al. [1501] show that, under the assumptions of their result, a neural network with only a small fixed number of stochastic variables can remain a universal approximator of conditional distributions. Restricting stochasticity therefore need not reduce predictive expressivity in principle. The result is existential, however, and does not prescribe how the stochastic parameters should be selected in practice.

The argument combines two established results. The first is the *noise outsourcing lemma* [64, 1874], which states that a conditional distribution can be represented through a measurable function that maps the input and an independent noise variable to the output. Consequently, samples from the conditional distribution can be obtained by sampling the noise variable and pushing it through this function. The corresponding function is sometimes called a *generator function*; its representation is generally not unique.

The second ingredient is a universal approximation result for feedforward neural networks [1014]. Combining these results yields the universal conditional-distribution approximation result of Sharma et al. [1501]. In particular, sufficiently expressive networks can approximate arbitrary conditional distributions while using only a small, fixed number of stochastic variables, provided that these variables enter the architecture before at least one nonlinear hidden layer.

A crucial feature of this construction is therefore that the stochastic variables are transformed by a nonlinear part of the network. Consequently, last-layer-only constructions do not possess the same universal conditional-distribution approximation property under this argument.

48.2.2 Subnetwork-based Laplace approximation

The theoretical result of Sharma et al. [1501] does not prescribe how to construct a semi-stochastic BNN in practice. It establishes that a small number of stochastic variables can be sufficient, but does not specify which network parameters should be treated stochastically.

Daxberger et al. [381] propose a practical heuristic for selecting these parameters. After fitting the full network by maximum a posteriori (MAP) estimation, they use a diagonal Laplace approximation to estimate the marginal posterior variances of the parameters. The parameters with the largest estimated variances are selected for the stochastic subnetwork. A Laplace approximation is then constructed over this restricted collection of parameters using the corresponding curvature information, while all remaining parameters are fixed at their MAP estimates.

The diagonal approximation is computationally attractive because its inverse can be obtained elementwise. It therefore provides an inexpensive ranking of the approximate marginal parameter variances without first constructing a full posterior approximation.

Other selection strategies are possible. For example, one may choose the parameters whose MAP estimates have the largest magnitudes, similarly to magnitude-based pruning. More sophisticated methods can instead select parameters or curvature entries using targeted subsampling of the Hessian matrix [503].

48.3 Subspaces of connected modes

In contrast to subnetwork inference, which fixes most coordinates in Θ and treats only a selected subset as stochastic around a point estimate, the approaches considered in this section exploit *mode connectivity*. Mode connectivity refers to the empirical observation that high-performing solutions can often be joined by continuous low-loss curves in parameter space [825].

The first step is to construct a low-dimensional connected parameterisation that follows such a curve. Optionally, the curve may be extended to a small ambient “tunnel” that permits posterior exploration in directions

normal to the path. Such constructions can capture richer predictive variation than an axis-aligned subnetwork of comparable dimension, although finding and representing the connected region is generally more expensive.

Bayesian inference is subsequently performed over the coordinates of the constructed parameterisation, while all remaining degrees of freedom are kept fixed.

48.3.1 Constructing a subspace

Let $T = [0, 1]$. A *loss path* is a continuous map $b : T \rightarrow \Theta$, evaluated through

$$\rho(t) = \ell(b(t)),$$

such that the loss remains comparatively small along the path. The map b may be represented using any sufficiently flexible family of curves. A well-performing choice is a regular Bézier curve with control points $\Lambda = \{\theta_0, \dots, \theta_K\}$ and Bernstein weights $\omega_k(t)$:

$$b_\Lambda(t) = \sum_{k=0}^K \omega_k(t) \theta_k, \quad \omega_k(t) = \binom{K}{k} (1-t)^{K-k} t^k. \quad (48.3)$$

A practical training objective is the path integral

$$\mathcal{L}(\Lambda) = \int_0^1 \ell(b_\Lambda(t)) dt = \mathbb{E}_{t \sim \text{Unif}(0,1)} [\ell(b_\Lambda(t))]. \quad (48.4)$$

This objective weights all values of the curve parameter t equally. It does not, in general, weight points uniformly with respect to arc length, since the speed $\|b'_\Lambda(t)\|$ may vary along the curve. A reparameterisation-invariant alternative is the arc-length-weighted objective

$$\tilde{\mathcal{L}}(\Lambda) = \frac{\int_0^1 \ell(b_\Lambda(t)) \|b'_\Lambda(t)\| dt}{\int_0^1 \|b'_\Lambda(t)\| dt}. \quad (48.5)$$

In practice, both objectives have been observed to produce similar results [436].

Optimising Equation 48.4 is straightforward. Let $\mathcal{I} \subseteq \{0, \dots, K\}$ denote the indices of the trainable control points. For each $k \in \mathcal{I}$, the gradient is

$$\begin{aligned} \nabla_{\theta_k} \mathcal{L} &= \mathbb{E}_{t \sim \text{Unif}(0,1)} \left[\left. \frac{\partial b_\Lambda(t)}{\partial \theta_k}^\top \nabla_{\vartheta} \ell(\vartheta) \right|_{\vartheta=b_\Lambda(t)} \right] \\ &= \mathbb{E}_{t \sim \text{Unif}(0,1)} \left[\omega_k(t) \nabla_{\vartheta} \ell(\vartheta) \Big|_{\vartheta=b_\Lambda(t)} \right]. \end{aligned} \quad (48.6)$$

Drawing a single value $t^* \sim \text{Unif}(0, 1)$ gives the unbiased single-sample estimator

$$\widehat{\nabla_{\theta_k} \mathcal{L}} = \omega_k(t^*) \nabla_{\vartheta} \ell(\vartheta) \Big|_{\vartheta=b_\Lambda(t^*)}. \quad (48.7)$$

The resulting gradient is the usual network gradient evaluated at $b_\Lambda(t^*)$, multiplied by the corresponding Bernstein weight. When the endpoints are fixed at two previously obtained modes, only the intermediate control points are optimised, so that $\mathcal{I} = \{1, \dots, K-1\}$. In mini-batch optimisation, drawing one value of t^* per iteration is often sufficient to approximate the expectation in Equation 48.4.

From path to tunnel To permit structured deviations from the curve, suppose that $b'_\Lambda(t) \neq 0$ and define the unit tangent

$$\tau(t) = \frac{b'_\Lambda(t)}{\|b'_\Lambda(t)\|}.$$

At each t , complete this tangent to an orthonormal frame

$$\{\tau(t), \nu_1(t), \dots, \nu_r(t)\},$$

where $\nu_1(t), \dots, \nu_r(t)$ are normal directions. A numerically stable choice is a rotation-minimising frame, which transports an initial normal frame along the curve while introducing as little additional rotation as possible. This avoids abrupt curvature-induced changes in the normal directions.

The resulting *tunnel* is parameterised by

$$\Phi = \left\{ g(t, \xi) = b_\Lambda(t) + \sum_{j=1}^r \xi_j \nu_j(t) : t \in [0, 1], \xi \in \Xi \subset \mathbb{R}^r \right\}. \quad (48.8)$$

This construction defines a connected $(1 + r)$ -dimensional parameterised region within Θ . We use the term *subspace* here in the broader sense common in the subspace-inference literature; mathematically, Φ is generally a nonlinear manifold or parameterised subset rather than a linear subspace. The set Ξ must be chosen sufficiently small, or optimised explicitly, to ensure that the tunnel remains within an acceptably low-loss neighbourhood of the original path.

48.3.2 Subspace posterior inference

Assume that a reduced parameterisation Φ has been constructed. The remaining degrees of freedom are fixed at a reference value $\hat{\theta}$, either explicitly or implicitly through the map g , and Bayesian inference is performed over the coordinates (t, ξ) .

For the normal coordinates, one possible prior is

$$p(\xi) = \mathcal{N}(0, \sigma^2 I),$$

possibly truncated to Ξ , with σ chosen relative to the tunnel radius. For the path coordinate, two natural but distinct choices are a uniform prior over the parameter,

$$t \sim \text{Unif}(0, 1),$$

and a prior that is uniform with respect to arc length. Let

$$s(t) = \int_0^t \|b'_\Lambda(u)\| du, \quad S = s(1).$$

A prior that is uniform in $s \in [0, S]$ induces the density

$$p(t) = \frac{\|b'_\Lambda(t)\|}{S}. \quad (48.9)$$

The two choices coincide only when the curve has constant speed.

Given a prior $p(t, \xi)$ on the coordinates, the posterior target is

$$p(t, \xi \mid \mathcal{D}) \propto p(\mathcal{D} \mid g(t, \xi)) p(t, \xi). \quad (48.10)$$

Here, $p(t, \xi)$ is interpreted as a density directly over the selected coordinates. If a density with respect to the volume measure induced on Φ is intended instead, the corresponding Jacobian determinant of g must be included.

The posterior can be explored using low-dimensional MCMC methods, such as the No-U-Turn Sampler (NUTS). Elliptical slice sampling is another option when the prior over the subspace coordinates has the required Gaussian structure. Reducing the dimension of the parameter space can make such classical sampling methods substantially easier to apply.

The reduced representation can also be combined with other structured models, facilitating the integration of deep neural networks into broader structured learning systems [435], including the use in large language models [437].

48.4 Related methods

Many other methods construct low-dimensional parameterisations or perform inference in reduced parameter spaces. Common examples exploit the geometry of optimiser trajectories.

Stochastic Weight Averaging (SWA) [SWA; 824] averages parameter snapshots from the late stages of stochastic gradient descent (SGD) to obtain a single, typically flatter solution that can improve generalisation and calibration. Stochastic Weight Averaging-Gaussian (SWAG) [1116] extends this idea by fitting a low-rank-plus-diagonal Gaussian distribution to the trajectory, thereby enabling parameter sampling and approximate Bayesian model averaging.

Another possibility is to apply principal component analysis (PCA) to a gradient-descent (GD) or SGD trajectory. PCA identifies dominant directions of variation in the optimiser iterates, which span a data-adaptive linear subspace. Bayesian inference can then be performed only over the coordinates associated with these directions.

Subspace inference can substantially reduce the computational cost of Bayesian neural network inference, but its performance depends strongly on how the reduced parameterisation is constructed. Axis-aligned subnetworks are comparatively simple, whereas mode-connected parameterisations can represent richer posterior variation at the cost of more involved optimisation and prior specification. Empirical evaluations should therefore assess not only predictive performance, but also sensitivity to the selected subspace, its dimension, and the prior placed over its coordinates.

Chapter 49

Implicitly Bayesian prediction rules

To reap the advantages of a Bayesian approach, most of this book prescribes a structured Bayesian approach: defining a parametric model, setting priors (Part VI), and approximating posteriors and posterior predictive distributions (Parts I, II, and III). But is that the only way? In this chapter, we will show how to characterise what makes a method Bayesian at a *behavioural* level, and illustrate that algorithms without an explicit prior-then-posterior construction can, in effect, be equally Bayesian.

Before answering this question, it is worth asking why we want to be Bayesian in the first place.

First and foremost, the Bayesian framework is often said to handle uncertainty in a principled manner. In machine learning, we are often concerned with making decisions based on predictions for future observations. With a Bayesian approach, that prediction takes the form of a full predictive probability distribution over the range of possible outcomes. Moreover, that distribution captures uncertainty in what the best possible distribution in the chosen model class might be: the Bayesian approach integrates over all possible predictions within the model class, weighting them according to their congruence with the data observed so far. Incorporating this uncertainty into the predictive distribution means that we can make decisions which minimise expected risk with respect to all the unknowns represented within the model. Bayesian decision theory guarantees that these decisions will be optimal under the (rather strict) assumption that our modelling choices are correct [123].

In practice, our prior and likelihood are rarely literal descriptions of either our beliefs or the data-generating process. However, even if the assumptions are wrong, Bayesian predictions retain a certain coherence property: whatever the stated beliefs about the state of the world were at the start, they will be updated rationally in the light of new data. Dutch book arguments give this property an operational interpretation, showing that incoherent updates expose an agent who acts on them to systematic exploitation [970, 1320]. This gives Bayesian predictions a measure of credibility, even when the modelling assumptions are wrong. For instance, when following the Bayesian approach exactly, one does not have to worry (as much) about whether some evidence is being given more weight than another, or whether some evidence is ignored outright. In the i.i.d. setting, for example, one might be concerned with whether a prediction depends arbitrarily on the order the data was presented in. Such order-invariance is not automatic in many approaches to prediction, such as deep learning: the order in which data are presented is known to significantly affect the final predictions [53, 1157]. Following the Bayesian approach prevents this. Even putting computational and approximate-inference considerations aside, a Bayesian still has plenty to worry about with regard to modelling choices; nonetheless, at the very least, some fundamental sanity checks are taken care of.

The conventional way to obtain these desirable predictive properties is through an explicit Bayesian construction: specify a prior, explicitly form a (typically approximate) posterior, then integrate it out to obtain predictions. In deep learning, however, this construction remains challenging. Algorithms that explicitly approximate posteriors in such Bayesian neural networks (BNNs) tend to encounter scalability challenges, which are the subject of much of this book. For instance, Markov chain Monte Carlo (MCMC) schemes (Part I) require, in effect, multiple models to be trained for an accurate approximation. Variational inference schemes, on the other hand, often struggle to fit complex posteriors over the parameters without similarly complex and difficult-to-train models to approximate that posterior.¹

These difficulties motivate asking whether the same predictive properties can be obtained without an explicit prior-posterior construction. The key observation is that coherence is a property of the predictions themselves, rather than of the mechanism through which they were computed. We therefore take a look at a different perspective on how to incorporate Bayesian principles into deep learning. In this chapter, we characterise the minimal

¹Function-space variational inference (covered in Chapter 17), however, offers a promising path towards resolving these issues.

conditions under which a prediction rule behaves as if it arose from an explicit Bayesian construction. We argue that this can be a more fruitful way to incorporate Bayesian principles into deep learning, as it only dictates how predictions should behave, rather than constraining how the prediction algorithm is structured. This greater flexibility suggests exploring novel approaches to making an algorithm more Bayesian, e.g. through better hyperparameter choices, explicit penalties, or pretraining techniques [583], without explicitly specifying a prior or approximating a posterior.

We will start this chapter by introducing the i.i.d. prediction setting and recapitulating what constitutes an explicitly Bayesian approach. Then, we will show what characterises the Bayesian approach at the level of predictions. We will give a construction for turning any finite-horizon prediction rule in this setting into one that behaves like a Bayesian rule, and show two ways in which the Bayesian counterpart is always strictly superior to the non-Bayesian one. Lastly, we will discuss how one can go about measuring how close any arbitrary predictor is to behaving like a Bayesian algorithm, and present this as a useful way for evaluating the fidelity of approximate inference for predictive approaches where explicit prior and posteriors are not trivially available (e.g. neural processes [1200]).

49.1 Explicitly vs. Implicitly Bayesian Methods

Whereas statistics often deals broadly with inferences about various unobserved quantities, the prediction of future observations and decision-making based on those predictions are arguably at the core of machine learning. We now make this sequential prediction setting precise. Given a sequence of n observations $X_1, \dots, X_n$, we are interested in predicting the value of X_{n+1} , for any possible n . We'll describe below how one would go about defining a prediction rule in the Bayesian framework in an i.i.d. context — namely, when the observations X_i are assumed to be independent and follow the same (data-generating) law $X_i \sim P_X$ — and then look at various properties that such prediction rules might have. We will restrict the exposition to categorical random variables; namely, the variables $X_1, X_2, \dots$ will take on values in some finite set $\mathcal{X}$. This greatly simplifies the mathematical probabilistic machinery that has to be deployed, but all the concepts readily generalise to more complex spaces (e.g. Euclidean with the usual σ -algebras) [1192].

To discuss and compare the properties of various rules for predicting future observations, it is helpful to precisely define what a *prediction rule* is. If the observations take values in some space $\mathcal{X}$, then a prediction rule is a sequence of functions $(s_0, s_1, s_2, \dots)$ where each s_n maps a finite sequence $(x_1, \dots, x_n) \in \mathcal{X}^n$ of previous observations to a prediction for the next observation. The prediction is probabilistic; namely, each $x_1, \dots, x_n$ maps to a *probability distribution* $s_n(\cdot | x_1, \dots, x_n)$ on $\mathcal{X}$. $s_n(\cdot | x_1, \dots, x_n)$ carries the interpretation of the prediction for the next observation X_{n+1} given the outcomes $(x_1, \dots, x_n)$ for the previous observations $X_1, \dots, X_n$. The following definition encapsulates the requirements:

Definition 49.1. Prediction rule (discrete) A sequence $(s_0, s_1, \dots)$ of functions $s_n : \mathcal{X}^{n+1} \rightarrow [0, 1]$ is said to be a (discrete) prediction rule if, for every $n \in \mathbb{N}$ and $(x_1, \dots, x_n) \in \mathcal{X}^n$, the mapping $x \mapsto s_n(x | x_1, \dots, x_n)$ is a probability mass function on $\mathcal{X}$. Namely, we require that $s_n(x | x_1, \dots, x_n) \in [0, 1]$ for all $x \in \mathcal{X}$, and $\sum_{x \in \mathcal{X}} s_n(x | x_1, \dots, x_n) = 1$.

A prediction rule is intended to describe how a practitioner makes predictions about the future observations given the past. In the context of deep learning, a prediction rule might encompass the whole procedure for training a neural network on a dataset of past observations, and then using the trained neural network to make predictions about future observations. For example, in the language modelling case where the examples x_i are sequences of discrete tokens, $s_n(x_{n+1} | x_1, \dots, x_n)$ might be defined as the probability assigned to the sequence x_{n+1} given by an autoregressive model trained on a dataset of examples $(x_1, \dots, x_n)$ following Stochastic Gradient Descent (SGD) with a maximum likelihood objective.²

The prediction rule $(s_0, s_1, \dots)$ gives us a way to assign probabilities (predictions) to the sequence of next n observations by defining a joint distribution $s_{1:n}$ like so:

$$s_{1:n}(x_1, \dots, x_n) \stackrel{\text{def}}{=} \prod_{i=1}^n s_{i-1}(x_i | x_1, \dots, x_{i-1}). \quad (49.1)$$

In fact, the prediction rule implies a (unique) probability measure on the entire sequence of possible future

²The sources of randomness for parameter initialisation and mini-batching would have to be fixed to yield a deterministic prediction rule.

observations $(x_1, x_2, x_3, \dots)$.³ In this sense, it can be interpreted as an *a priori* prediction for what might happen, even before we observe any data.

Vice versa, we can obtain a prediction rule from any probability measure s over infinite sequences $\mathcal{X}^\infty$ in a natural way by marginalising out to get the conditional of interest. Concretely, if we define a sequence of random variables $(\bar{X}_1, \bar{X}_2, \dots)$ (on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$) to follow the joint law s , we can specify a prediction rule with a sequence of prediction functions $s_n(x_{n+1}|x_1, \dots, x_n) \stackrel{\text{def}}{=} \mathbb{P}[\bar{X}_{n+1} = x_{n+1} | \bar{X}_1 = x_1, \dots, \bar{X}_n = x_n]$. This prediction rule will in turn imply the same joint distribution s on the infinite sequence space. This equivalence between prediction rules and joints over infinite sequences of observations allows us to define properties of prediction rules in terms of the joint they imply over the entire sequence space.

49.1.1 The Bayesian Inferential Approach

In this section, we will go through how we could construct a ‘traditionally’ Bayesian prediction rule. A prototypical prescription for inference following the Bayesian framework involves: **1)** specifying a joint distribution over all the variables of interest, and then **2)** conditioning on the observed values to obtain a posterior distribution over the unobserved variables of interest. In the context of a sequential prediction problem, one might specify a joint distribution over a sequence of observations in $\mathcal{X}^\infty$, treating the observations as random variables $(X_1, X_2, \dots)$. Then, one would condition on the observed values of $X_1, \dots, X_n$ to obtain a posterior distribution for e.g. X_{n+1} .

In the context of i.i.d. data — where the true observations $(X_1, X_2, \dots)$ are assumed to be independent and identically distributed following the same law $X_i \sim P_X$ — a Bayesian would usually treat the data generating distribution P_X as an unknown, and place a prior on it. To make predictions about the next observation X_{n+1} given observed outcomes $(x_1, \dots, x_n)$ for the previous observations $X_1, \dots, X_n$, one could describe the inferential (explicitly) Bayesian approach as:⁴

- I. Specify a prior over the generating distribution;
- II. Get a posterior over the generating distribution given the observed data $x_1, \dots, x_n$;
- III. Compute the prediction $s_n(\cdot | x_1, \dots, x_n)$ by computing the posterior predictive distribution for X_{n+1} .

For instance, suppose the distributions of interest are described by an unknown parameter Θ , treated as a random variable taking values in $\mathbb{R}^d$, with likelihood $p_{X|\Theta}(x|\theta)$ and prior density $p_\Theta(\theta)$.⁵ The procedure above might then look like:

- I. Specify a *likelihood* $p_{X|\Theta}(x|\theta)$ and a *prior* density $p_\Theta(\theta)$ over the parameters;
- II. Compute the posterior density $p_{\Theta|X_1, \dots, X_n}(\theta | x_1, \dots, x_n) \propto \prod_{i=1}^n p_{X|\Theta}(x_i|\theta) p_\Theta(\theta)$;
- III. Compute the prediction $s_n(\cdot | x_1, \dots, x_n) \stackrel{\text{def}}{=} \int p_{X|\Theta}(\cdot|\theta) p_{\Theta|X_1, \dots, X_n}(\theta | x_1, \dots, x_n) d\theta$.

In the next section, we will see this is not the only way to obtain prediction rules that are Bayesian.

49.1.2 Implicitly Bayesian Prediction Rules

The explicit computation of the posterior over the parameters can, in many cases, be side-stepped.

For example, consider a Bayesian model for a sequence of paired observations $((X_1, Y_1), (X_2, Y_2), \dots)$, where X_n, Y_n are random variables on $\mathbb{R}^d$ and $\mathbb{R}$ respectively (briefly deviating from the categorical setting). We could specify a Bayesian linear model on (X_n, Y_n) , $Y_n = \theta^\top X_n + \varepsilon_n$, with a uniform density on the covariates X_n over $[0, 1]^d$, a Gaussian prior $\mathcal{N}(\theta; 0, I_d)$ on the weights $\theta \in \mathbb{R}^d$, and Gaussian observation noise $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$. For $x_{n+1} \in [0, 1]^d$, the posterior predictive density for (X_{n+1}, Y_{n+1}) taking the value (x_{n+1}, y_{n+1}) , conditioned on the observed values $((X_1, Y_1), \dots, (X_n, Y_n)) = ((x_1, y_1), \dots, (x_n, y_n))$, can then be written as:

$$\begin{aligned} s_n((x_{n+1}, y_{n+1}) | (x_1, y_1), \dots, (x_n, y_n)) &= \mathcal{N}(y_{n+1}; \bar{\mu}, \bar{\sigma}^2) \\ \bar{\mu} &= x_{n+1}^\top \Phi^\top (\Phi \Phi^\top + \sigma^2 I_n)^{-1} y_{1:n}, \\ \bar{\sigma}^2 &= x_{n+1}^\top x_{n+1} + \sigma^2 - x_{n+1}^\top \Phi^\top (\Phi \Phi^\top + \sigma^2 I_n)^{-1} \Phi x_{n+1}, \end{aligned}$$

³Note that this will be a distribution over infinite sequences. In the discrete case, the existence follows from Carathéodory’s extension theorem [157].

⁴The “inferential approach” terminology was taken from Berti et al. [138], who distinguish between an explicit prior–posterior construction (the *inferential approach*), and the direct specification of predictive distributions (the *predictive approach*).

⁵We write $p_X(x)$ for the density or probability mass function of a random variable X evaluated at a realisation x , and $p_{X|Y}(x|y)$ for its conditional counterpart given $Y = y$.

with $\Phi = [x_1 \ \dots \ x_n]^\top \in \mathbb{R}^{n \times d}$ and $y_{1:n} = [y_1 \ \dots \ y_n]^\top$. The uniform density of X_{n+1} equals one on $[0, 1]^d$, so its factor disappears from the predictive density shown above.

Following this prediction rule, the practitioner never explicitly *manifests* a posterior over the weights. Nonetheless, they are, of course, acting (making predictions) as *if* they were computing and then marginalising out the posterior. The predictions are the same. In this sense, the prediction rule could be said to be *behaviourally* or *implicitly* Bayesian. The predictive perspective has a long history in statistics [377, 593, 1412]. More recent work has developed direct specifications of prediction rules [133, 135–137, 538, 539], often referred to as the Bayesian *predictive approach* [138].

If prediction rules can behave as if they were following a Bayesian approach, a natural question to ask is: under what conditions is a prediction rule equivalent to an explicitly Bayesian approach for *some* prior and likelihood? In other words, given a prediction rule, can one say whether there exists a prior and likelihood such that the predictions from the prediction rule match those of following the Bayesian framework with that likelihood/prior pair under the i.i.d. assumption? If this is the case, we'll say that the prediction rule is *implicitly Bayesian*. The prediction rule and the corresponding likelihood/prior construction can then be said to be *Bayes-ically* the same.

Implications for Bayesian Deep Learning The fact that prediction rules can be Bayesian without explicitly being constructed as such opens some new avenues for Bayesian deep learning research. It suggests that we could construct predictors in a way that's convenient, with complete disregard for Bayesian principles, and then measure or optimise for how close to being *implicitly* Bayesian they are. For this, we would need some notion of a measure of implicit Bayesianity, which we discuss in later sections. Alternatively, perhaps we could take a non-Bayesian prediction rule, and turn it into a Bayesian one post hoc? For either of those, it is useful to know what actually makes a prediction rule Bayesian. In what follows, we'll look at the properties that the prediction rules defined following the Bayesian framework possess, and the properties that characterise them.

49.2 Characterising Implicitly Bayesian Prediction Rules

In the case of Bayesian models assuming i.i.d. data, de Finetti's theorem gives a simple condition for a prediction rule to be implicitly Bayesian. As mentioned before, a prediction rule implies a unique joint distribution over a sequence of observations $(X_1, X_2, \dots)$. By a version of de Finetti's theorem, under some mild assumptions, a prediction rule is implicitly Bayesian if and only if the joint distribution it implies over $(X_1, X_2, \dots)$ is *exchangeable* [745]:

Definition 49.2 (Exchangeable Sequence of Random Variables). *A finite sequence of n random variables $(X_1, \dots, X_n)$ is said to be exchangeable if for any permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ the joint distribution of $(X_1, \dots, X_n)$ is the same as the joint distribution of $(X_{\pi(1)}, \dots, X_{\pi(n)})$:*

$$\mathbb{P}[X_1 = x_1, \dots, X_n = x_n] = \mathbb{P}[X_{\pi(1)} = x_1, \dots, X_{\pi(n)} = x_n] \quad \forall (x_1, \dots, x_n) \in \mathcal{X}^n$$

An infinite sequence of random variables $(X_1, X_2, \dots)$ is said to be exchangeable if, for any n , the finite sequence $(X_1, \dots, X_n)$ is exchangeable.

In the binary case, de Finetti's theorem can be stated as:

Theorem 49.1 (de Finetti's theorem). *Consider an infinite sequence of random variables $(X_1, X_2, \dots)$ on $\mathcal{X} = \{0, 1\}$. Then, the following are equivalent:*

1. *There exists a probability measure π on $[0, 1]$ (with the usual σ -algebra) such that:*

$$\mathbb{P}[X_1 = x_1, \dots, X_n = x_n] = \int \prod_{i=1}^n \theta^{x_i} (1 - \theta)^{1-x_i} d\pi(\theta) \quad \forall x_1, \dots, x_n \in \{0, 1\} \forall n \in \mathbb{N}$$

2. *The sequence $(X_1, X_2, \dots)$ is exchangeable.*

Of course, condition 1. above implies that the predictions $s_n(x|x_1, \dots, x_n) = \mathbb{P}[X_{n+1} = x|X_1 = x_1, \dots, X_n = x_n]$ are made by computing and marginalising a posterior:

$$\begin{aligned} \mathbb{P}[X_{n+1} = x|X_1 = x_1, \dots, X_n = x_n] &\stackrel{(\text{in } x)}{\propto} \mathbb{P}[X_1 = x_1, \dots, X_n = x_n, X_{n+1} = x] \\ &\propto \underbrace{\int \theta^x (1 - \theta)^{1-x} d\pi(\theta)}_{\text{likelihood}} \underbrace{\prod_{i=1}^n \theta^{x_i} (1 - \theta)^{1-x_i} d\pi(\theta)}_{\propto \text{posterior}} \end{aligned}$$

Hence, exchangeability is the defining characteristic of implicitly Bayesian prediction rules on i.i.d. data. This suggests one direct way of checking whether a prediction rule is implicitly Bayesian: check whether the joint distribution on $(X_1, X_2, \dots)$ implied by the prediction rule is exchangeable.

49.2.1 Conditionally Identically Distributed

Another desirable property that we might expect of a prediction rule is that the future observations are identically distributed given the past. For example, under the i.i.d. assumption, if we were to observe $(x_1, \dots, x_n)$, the prediction for the next observation X_{n+1} *surely* shouldn't be different from the prediction for the observation after that. After all, we know they are identically distributed, we just don't know what the distribution is; it'd be irrational to make different predictions for X_{n+1}, X_{n+2} given the same data. This property can be formalised as follows:

Definition 49.3 (Conditionally Identically Distributed). *We say that a sequence of random variables $(X_1, X_2, \dots)$ is conditionally identically distributed (c.i.d.) if for any n and any $(x_1, \dots, x_n)$ with a positive probability:*

$$\mathbb{P}[X_{n+1} = x | X_1 = x_1, \dots, X_n = x_n] = \mathbb{P}[X_{n+k} = x | X_1 = x_1, \dots, X_n = x_n] \quad \forall k > 0$$

holds almost surely.

A prediction rule is then *c.i.d.* if the joint distribution it implies over $(X_1, X_2, \dots)$ is *c.i.d.* *c.i.d.* sequences were introduced in [861] and studied and applied in a range of works [134, 135, 540].

It should be clear that each exchangeable sequence is *c.i.d.*, as exchangeability implies $X_1, \dots, X_n, X_{n+1} \stackrel{d}{=} X_1, \dots, X_n, X_{n+k}$ for any $k > 0$. Hence, the Bayesian framework does yield *c.i.d.* prediction rules, and any implicitly Bayesian prediction rule will be *c.i.d.* Not all *c.i.d.* sequences are exchangeable, however. *c.i.d.* is a weaker condition than exchangeability.

49.2.2 Stationarity

One property that's missing from just *c.i.d.* sequences that would make them *implicitly Bayesian* is *stationarity*:

Definition 49.4 (Stationary sequence). *A sequence of random variables $(X_1, X_2, \dots)$ is said to be stationary if for any $n, k > 0$:*

$$X_1, \dots, X_n \stackrel{d}{=} X_{1+k}, \dots, X_{n+k}, \quad (49.2)$$

where $\stackrel{d}{=}$ denotes equality in distribution.

In the context of prediction rules, stationarity implies sampling k datapoints $x'_1, \dots, x'_k$ from the prediction rule itself and prepending them to the observed data $\mathcal{D} = (x_1, \dots, x_n)$ would yield a prediction rule $s_{n+k}(\cdot | x'_1, \dots, x'_k, \mathcal{D})$ that's in expectation the same as $s_n(\cdot | x_1, \dots, x_n)$.

By the result of Kallenberg [861, Proposition 2.1], exchangeability exactly amounts to stationarity and the *c.i.d.* condition; stationarity and *c.i.d.* properties are another way of characterising implicitly Bayesian prediction rules.

49.2.3 Spreadability

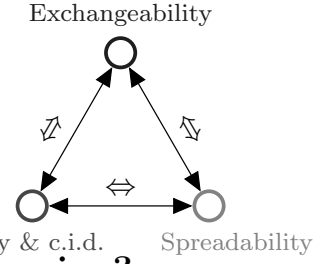
Another condition that turns out to also characterise implicitly Bayesian models is *spreadability*:

Definition 49.5 (Spreadable sequence). *A sequence of random variables $(X_1, X_2, \dots)$ is said to be spreadable if for any $n > 0$ and any sequence of indices $k_n > k_{n-1} > \dots > k_1 \geq 1$:*

$$X_1, \dots, X_n \stackrel{d}{=} X_{k_1}, \dots, X_{k_n} \quad (49.3)$$

Spreadability says that the distribution of the first n observations is the same as the distribution of any n observations from the sequence. In the context of a prediction rule, it implies that, if we were to construct a new dataset by possibly sampling $k_i \geq 0$ datapoints from the prediction rule conditioned on the previous datapoints in-between "observed" datapoints x_i, x_{i+1} , the final prediction given that dataset would in expectation be the same as the prediction given the original dataset. Kallenberg [861] has shown that spreadability is equivalent to exchangeability. In other words, spreadability is yet another property that characterises implicitly Bayesian prediction rules.

In summary, an implicitly Bayesian prediction rule can be equivalently characterised by *exchangeability*, *spreadability*, or *stationarity* together with the *c.i.d.* condition. Some of these properties might be easier to establish or enforce than others, and each suggests its own way of constructing an implicitly Bayesian prediction rule from a non-Bayesian one. In the next section, we will illustrate how this can be done for the *exchangeability* criterion, and what benefits it can bring.



49.3 Why and how to make predictors implicitly Bayesian?

The equivalence between implicitly Bayesian predictors and “ordinary” Bayesian predictors might be appealing to some readers, but why would a practitioner that has a prediction rule they enjoy care that it’s not implicitly Bayesian? If it ‘works well’ on performance benchmarks, what does it matter? In this section, we’ll show that algorithms that are not implicitly Bayesian (ones that lack exchangeability at a fixed horizon) leave some performance on the table. In fact, any non-implicitly Bayesian prediction rule, once restricted to prediction over a finite horizon, has a finitely exchangeable counterpart whose predictive performance is strictly better. Below, we will illustrate a specific approach for taking an arbitrary non-Bayesian prediction rule and turning it into an implicitly Bayesian one.⁶ This can be of independent interest as a basis for a practical algorithm, but it will also serve us as a tool for showing non-Bayesian predictors always leave something on the table compared to Bayesian ones.

49.3.1 Turning a prediction rule Bayesian

Consider an arbitrary prediction rule $(s_0, \dots, s_{n-1})$ restricted to a finite horizon $n \in \mathbb{N}$. By Equation (49.1), this implies a joint $s_{1:n}$ on the finite sequence space $\mathcal{X}^n$. We can then define a projection operator Exch_n that takes a joint on $\mathcal{X}^n$, and turns it into a finitely exchangeable joint distribution $\hat{s}_{1:n}$ by averaging over the set Π_n of all possible permutations of $\{1, \dots, n\}$:

$$\hat{s}_{1:n} = \text{Exch}_n(s_{1:n}) \quad \Leftrightarrow \quad \hat{s}_{1:n}(x_1, \dots, x_n) = \frac{1}{n!} \sum_{\pi \in \Pi_n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}). \quad (49.4)$$

This is a projection, as it will leave an already exchangeable joint unaltered. Similarly, the resulting joint is itself permutation-invariant in its arguments, and hence finitely exchangeable. Finally, from the new joint, we can recover a new finite-horizon prediction rule $(\hat{s}_0, \dots, \hat{s}_{n-1})$ by marginalising in the obvious way:

$$\hat{s}_0(x_1) = \sum_{x_2, \dots, x_n} \hat{s}_{1:n}(x_1, \dots, x_n) \quad \hat{s}_1(x_2|x_1) = \frac{1}{\hat{s}_0(x_1)} \sum_{x_3, \dots, x_n} \hat{s}_{1:n}(x_1, \dots, x_n) \quad \dots$$

49.3.2 Exchangeable-ified prediction rules are always better

Below, we show that, when both the original finite-horizon predictive distribution and the exchangeable data-generating distribution have full support, the rule obtained by applying the projection Exch_n is strictly better unless the original finite-horizon rule is already exchangeable (in which case $\text{Exch}_n(s_{1:n}) = s_{1:n}$).

Proposition 49.1. *Consider full-support probability mass functions $s_{1:n}$ and p on a finite sequence space $\mathcal{X}^n$, where p is exchangeable, and take $\hat{s}_{1:n} = \text{Exch}_n(s_{1:n})$. Then:*

$$D_{\text{KL}}(s_{1:n} \| p) = D_{\text{KL}}(\hat{s}_{1:n} \| p) + D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}), \quad (49.5)$$

and, in particular, $D_{\text{KL}}(s_{1:n} \| p) \geq D_{\text{KL}}(\hat{s}_{1:n} \| p)$ with equality if and only if $s_{1:n}$ is exchangeable.

⁶This will be done, in contrast to the previous sections, for predictions on finite sequences. To our knowledge, it’s an open problem as to whether similar constructions can be formulated for infinite sequences, and what the necessary preconditions might be.

Proof. We can expand $D_{\text{KL}}(s_{1:n} \| p)$ in the following way:

$$\begin{aligned} D_{\text{KL}}(s_{1:n} \| p) &= \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_1, \dots, x_n) \log \frac{s_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \\ &= \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_1, \dots, x_n) \left(\log \frac{s_{1:n}(x_1, \dots, x_n)}{\hat{s}_{1:n}(x_1, \dots, x_n)} + \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \right) \\ &= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_1, \dots, x_n) \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \end{aligned}$$

Introduce an average over all possible permutations Π_n

$$= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_1, \dots, x_n) \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)}$$

Applying a change of variables $(x_1, \dots, x_n) \mapsto (x_{\pi(1)}, \dots, x_{\pi(n)})$ to the inner summation:

$$= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \log \frac{\hat{s}_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)})}{p(x_{\pi(1)}, \dots, x_{\pi(n)})}$$

By exchangeability of $\hat{s}_{1:n}$ and p we can ‘unpermute’ the elements:

$$\begin{aligned} &= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \\ &= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \sum_{x_{1:n} \in \mathcal{X}^n} \left(\frac{1}{n!} \sum_{\pi \in \Pi_n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \right) \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \\ &= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + \sum_{x_{1:n} \in \mathcal{X}^n} \hat{s}_{1:n}(x_1, \dots, x_n) \log \frac{\hat{s}_{1:n}(x_1, \dots, x_n)}{p(x_1, \dots, x_n)} \\ &= D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n}) + D_{\text{KL}}(\hat{s}_{1:n} \| p) \end{aligned}$$

Clearly, by known properties of the KL-divergence, the gap $D_{\text{KL}}(s_{1:n} \| \hat{s}_{1:n})$ will be 0 if and only if the two distributions are the same. The distributions will only be the same if $s_{1:n}$ is already exchangeable. This completes the proof. $\square$

An equivalent claim can be made for the reverse KL-divergence:

Proposition 49.2. *Consider full-support probability mass functions $s_{1:n}$ and p on a finite sequence space $\mathcal{X}^n$, where p is exchangeable, and take $\hat{s}_{1:n} = \text{Exch}_n(s_{1:n})$. Then:*

$$D_{\text{KL}}(p \| s_{1:n}) \geq D_{\text{KL}}(p \| \hat{s}_{1:n}), \quad (49.6)$$

with equality if and only if $s_{1:n}$ is exchangeable.

Proof. The proof rests on applying Jensen’s inequality to the term:

$$\log \left(\frac{1}{n!} \sum_{\pi \in \Pi_n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \right) \geq \frac{1}{n!} \sum_{\pi \in \Pi_n} \log (s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}))$$

Concretely, by expanding $D_{\text{KL}}(p \| \hat{s}_{1:n})$ we get:

$$\begin{aligned} D_{\text{KL}}(p \| \hat{s}_{1:n}) &= \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) \log \frac{p(x_1, \dots, x_n)}{\hat{s}_{1:n}(x_1, \dots, x_n)} \\ &= \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) \left(\log p(x_1, \dots, x_n) - \log \left(\frac{1}{n!} \sum_{\pi \in \Pi_n} s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \right) \right) \end{aligned}$$

By Jensen's inequality (since $\log$ is concave, $\log(\mathbb{E}[X]) \geq \mathbb{E}[\log(X)]$):

$$\begin{aligned}
&\leq \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) \left(\log p(x_1, \dots, x_n) - \frac{1}{n!} \sum_{\pi \in \Pi_n} \log s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) \right) \\
&= \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) (\log p(x_1, \dots, x_n) - \log s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)})) \\
&= \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) \log \frac{p(x_1, \dots, x_n)}{s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)})}
\end{aligned}$$

Change of variables $(x_1, \dots, x_n) \mapsto (x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)})$ in the inner summation:

$$= \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} p(x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)}) \log \frac{p(x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)})}{s_{1:n}(x_1, \dots, x_n)}$$

By exchangeability of p , $p(x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)}) = p(x_1, \dots, x_n)$:

$$\begin{aligned}
&= \frac{1}{n!} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} p(x_1, \dots, x_n) \log \frac{p(x_1, \dots, x_n)}{s_{1:n}(x_1, \dots, x_n)} \\
&= \frac{1}{n!} \sum_{\pi \in \Pi_n} D_{\text{KL}}(p \| s_{1:n}) \\
&= D_{\text{KL}}(p \| s_{1:n})
\end{aligned}$$

For strictly concave functions (such as $\log$), Jensen's inequality becomes an equality only if the averaged terms are all the same. For the inequality above, this implies we get an equality if and only if:

$$s_{1:n}(x_{\pi(1)}, \dots, x_{\pi(n)}) = s_{1:n}(x_1, \dots, x_n),$$

for all $\pi \in \Pi_n$, which completes the proof of the proposition. $\square$

Related arguments in the broader setting of statistical decision theory go back to Wald [1693].

On the basis of these results, one can make a compelling case for being Bayesian without alluding to uncertainty quantification properties. Even if you only care about predictive performance, you would still be universally better off with *some* Bayesian algorithm, at least according to some metrics!

49.3.3 Non-implicitly Bayesian agents are vulnerable to adversarial bets

As a final argument in favour of Bayesian prediction rules, we show that by following non-Bayesian prediction rules, one is vulnerable to accepting seemingly favourable bets that will in expectation always be disadvantageous. An adversary can come up with such bets by, once again, utilising the construction in Equation (49.4), and exploiting any incoherence in the predictive distribution.⁷

Consider a betting game in which two players — an agent and an adversary — make bets on the next sequence of n observations, each observation taking value in some space $\mathcal{X}$. At the end, a transfer of funds happens between the agent and the adversary depending on the observed outcomes.

Formally, a *bet* is a function $r : \mathcal{X}^n \rightarrow \mathbb{R}$. The value of r can be thought of as representing the transfer of funds from the adversary to the agent for each possible outcome; i.e. if the observed outcome is $x_1, \dots, x_n$ and $r(x_1, \dots, x_n)$ is positive, the agent receives $r(x_1, \dots, x_n)$ from the adversary. If, on the other hand, the observed outcome is $x_1, \dots, x_n$ and $r(x_1, \dots, x_n)$ is negative, the agent gives $-r(x_1, \dots, x_n)$ to the adversary.

We will consider players that make predictions on the sequence of observations, and choose which bets to accept based on those beliefs. Concretely, *agent beliefs* and *adversary beliefs* are probability mass functions on $\mathcal{X}^n$.

Definition 49.6 (Bets favourable to agent). *Given agent beliefs $q : \mathcal{X}^n \rightarrow [0, 1]$, a function $r : \mathcal{X}^n \rightarrow \mathbb{R}$ is said to be a **bet favourable to the agent** if:*

$$\sum_{x_{1:n} \in \mathcal{X}^n} r(x_{1:n}) q(x_{1:n}) > 0. \quad (49.7)$$

⁷ A similar result could be shown to follow directly from Proposition 49.2, which would be akin to many known information-theoretic betting results.

In other words, a bet is favourable to the agent if the ‘expected return’ from the point of view of the agent is strictly positive.

Definition 49.7 (Bets favourable to adversary). *Given adversary beliefs $\bar{q} : \mathcal{X}^n \rightarrow [0, 1]$, a function $r : \mathcal{X}^n \rightarrow \mathbb{R}$ is said to be a **bet favourable to the adversary** if:*

$$\sum_{x_{1:n} \in \mathcal{X}^n} (-r(x_{1:n})) \bar{q}(x_{1:n}) > 0. \quad (49.8)$$

Definition 49.8 (Admissible bets). *Given agent and adversary beliefs, a bet r is said to be an **admissible bet** if it is simultaneously a bet favourable to the agent and a bet favourable to the adversary.*

Definition 49.9 (Minimal bets). *Given agent beliefs $q : \mathcal{X}^n \rightarrow [0, 1]$, and adversary beliefs $\bar{q} : \mathcal{X}^n \rightarrow [0, 1]$, a function $r : \mathcal{X}^n \rightarrow \mathbb{R}$ is said to be a **minimal bet** if it’s a linear combination of agent and adversary beliefs, namely:*

$$r(x_{1:n}) = aq(x_{1:n}) + b\bar{q}(x_{1:n}) \quad \text{for some } a, b \in \mathbb{R}. \quad (49.9)$$

The rationale for this is that adding components to the bet that are orthogonal to the beliefs of the agent and the adversary will not change the expected return of the bet for either of them. In other words, for any bet r that is not minimal, there exists a ‘smaller’ minimal bet r' such that the expected returns of r and r' are the same for both the agent and the adversary. Restricting ourselves to minimal bets hence simply ensures that the agents do not arbitrarily make bets that are not justified on the grounds of their beliefs.

Given a bet function $r : \mathcal{X}^n \rightarrow \mathbb{R}$, and a *true distribution* $p : \mathcal{X}^n \rightarrow [0, 1]$, the game proceeds by sampling observations $X_1, \dots, X_n$ from p — i.e. $(X_1, \dots, X_n) \sim p$ — and at the end of the game a transfer of funds occurs. The *reward to the agent* is defined as $r(X_1, \dots, X_n)$, and the *expected return to the agent* is: $\mathbb{E}[r((X_1, \dots, X_n))] = \sum_{x_{1:n} \in \mathcal{X}^n} p(x_{1:n})r(x_{1:n})$.

Definition 49.10 (ℓ_2 norm). *For functions $f : \mathcal{X}^n \rightarrow \mathbb{R}$, the ℓ_2 norm of f , written $\|f\|_2$, is defined as $\|f\|_2 = \sqrt{\sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n})^2}$.*

We will consider the case of an adversary that constructs an “exchangeable-ified” version of the agent’s beliefs: $\bar{q} = \text{Exch}_n(q)$ as per Equation (49.4). One can interpret this as a scenario in which the adversary doesn’t know the true probabilities of the events, but knows the beliefs of the agent, and so they are able to exploit a possible lack of exchangeability in the agent’s beliefs.

The theorem we will state below roughly says “if a sensible bet is made between the agent and the adversary, then *no matter what the true distribution is* the expected return for the agent is strictly negative”. The minimality of the bet is required for this to hold: if the bets are allowed to not be minimal, then we could find some true distribution that rewards one of the bettors despite them having no ‘edge’ or advantage in terms of their beliefs.

This brings us to the following result, which we will now prove:

Theorem 49.2. *Given an agent with beliefs $q : \mathcal{X}^n \rightarrow [0, 1]$, and an adversary with “exchangeable-ified” beliefs $\bar{q} = \text{Exch}_n(q)$ as defined in Equation (49.4), for any exchangeable true distribution $p : \mathcal{X}^n \rightarrow [0, 1]$ whose support overlaps that of q , all minimal and admissible bets r have strictly negative expected return for the agent under p :*

$$\sum_{x_{1:n} \in \mathcal{X}^n} p(x_{1:n})r(x_{1:n}) < 0 \quad (49.10)$$

Furthermore, either:

- q is exchangeable and $\bar{q} = q$, and there are no admissible bets.
- q is not exchangeable and there exist admissible (and minimal) bets.

In particular, Theorem 49.2 holds when p is an i.i.d. distribution.

Proof. A note on notation It is helpful to abbreviate the notation in parts of the proof by thinking of the agent and adversary beliefs, as well as bets, as vectors. If we enumerate all the elements of $\mathcal{X}^n$, we can represent functions from $\mathcal{X}^n$ to $\mathbb{R}$ as vectors in $\mathbb{R}^{|\mathcal{X}^n|}$. For example, if $\mathcal{X} = \{0, 1\}$, $n = 2$, and the chosen ordering of elements of $\mathcal{X}^n$ is $((0, 0), (0, 1), (1, 0), (1, 1))$, then the function r can be represented as a vector in $\mathbb{R}^4$: $r = [r(0, 0) \ r(0, 1) \ r(1, 0) \ r(1, 1)]^T$. In this proof, we will switch freely between the functions and vectors for the same function, as it should always be apparent from the context what the counterpart is.

For any function $f : \mathcal{X}^n \rightarrow \mathbb{R}$ that's permutation-invariant in its arguments we have that:

$$\begin{aligned} \mathbf{f} \cdot \bar{\mathbf{q}} &= \sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n}) \bar{q}(x_{1:n}) = \sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n}) \left(\frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} q(\pi(x_{1:n})) \right) \\ &= \frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n}) q(\pi(x_{1:n})) = \frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} f(\pi(x_{1:n})) q(\pi(x_{1:n})) \\ &= \frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} \sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n}) q(x_{1:n}) = \sum_{x_{1:n} \in \mathcal{X}^n} f(x_{1:n}) q(x_{1:n}) = \mathbf{f} \cdot \mathbf{q}, \end{aligned}$$

where the second to last equality holds because permuting the elements of both vectors in a dot product $\mathbf{f} \cdot \mathbf{q}$ doesn't change the result. Hence:

$$\mathbf{p} \cdot \bar{\mathbf{q}} = \mathbf{p} \cdot \mathbf{q} \qquad \bar{\mathbf{q}} \cdot \bar{\mathbf{q}} = \mathbf{q} \cdot \bar{\mathbf{q}} \quad (49.11)$$

I. All admissible and minimal bets have negative expected return. Suppose r is minimal and admissible. Then, by minimality, r can be represented as:

$$r = a\mathbf{q} + b\bar{\mathbf{q}}, \quad (49.12)$$

for some constants $a, b \in \mathbb{R}$. By admissibility of r , we can further get the following inequality for a, b :

$$\mathbf{q} \cdot r > 0 \qquad \Rightarrow a(\mathbf{q} \cdot \mathbf{q}) + b(\mathbf{q} \cdot \bar{\mathbf{q}}) > 0 \quad (49.13)$$

$$\begin{aligned} \bar{\mathbf{q}} \cdot r < 0 &\Rightarrow a(\bar{\mathbf{q}} \cdot \mathbf{q}) + b(\bar{\mathbf{q}} \cdot \bar{\mathbf{q}}) = a(\bar{\mathbf{q}} \cdot \mathbf{q}) + b(\bar{\mathbf{q}} \cdot \mathbf{q}) < 0 \\ &\Rightarrow a + b < 0. \end{aligned} \quad (49.14)$$

as $\mathbf{q} \cdot \bar{\mathbf{q}} > 0$. By (49.12), we can rewrite the expected return (for the agent) as:

$$\mathbf{p} \cdot r = a\mathbf{p} \cdot \mathbf{q} + b\mathbf{p} \cdot \bar{\mathbf{q}} = a\mathbf{p} \cdot \mathbf{q} + b\mathbf{p} \cdot \mathbf{q} = (a + b)(\mathbf{p} \cdot \mathbf{q}),$$

where the second to last equality follows by (49.11).

Now, $\mathbf{p} \cdot \mathbf{q} > 0$ as long as $\mathbf{p}$ and $\mathbf{q}$ have overlapping support, and by (49.14), $a + b < 0$, hence:

$$\mathbf{p} \cdot r < 0,$$

as required.

II. Minimal and admissible bets exist if and only if $\mathbf{q} \neq \bar{\mathbf{q}}$. Suppose that $\mathbf{q} = \bar{\mathbf{q}}$. An admissible bet must satisfy:

$$\begin{aligned} 0 &< r \cdot \mathbf{q} \\ 0 &> r \cdot \bar{\mathbf{q}} = r \cdot \mathbf{q}, \end{aligned}$$

which is a contradiction, and so $\mathbf{q} = \bar{\mathbf{q}}$ implies that no admissible bet exists.

Now, suppose that $\mathbf{q} \neq \bar{\mathbf{q}}$. Note that:

$$\begin{aligned} \|\bar{\mathbf{q}}\|_2^2 &= \sum_{x_{1:n} \in \mathcal{X}^n} \left(\frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} q(\pi(x_{1:n})) \right)^2 && \text{(apply Jensen's inequality)} \\ &\leq \sum_{x_{1:n} \in \mathcal{X}^n} \frac{1}{|\Pi_n|} \sum_{\pi \in \Pi_n} q(\pi(x_{1:n}))^2 = \sum_{x_{1:n} \in \mathcal{X}^n} q(x_{1:n})^2 = \|\mathbf{q}\|_2^2, \end{aligned}$$

and so, $\mathbf{q} \cdot \mathbf{q} > \mathbf{q} \cdot \bar{\mathbf{q}}$ (strict inequality, as $\mathbf{q} \neq \bar{\mathbf{q}}$). Hence, picking $a = \left(\frac{\mathbf{q} \cdot \bar{\mathbf{q}}}{\mathbf{q} \cdot \mathbf{q}} \right)^{\frac{1}{2}}$ and $b = -1$ yields an admissible and minimal bet $r = a\mathbf{q} + b\bar{\mathbf{q}}$, because:

$$\begin{aligned} a(\mathbf{q} \cdot \mathbf{q}) + b(\mathbf{q} \cdot \bar{\mathbf{q}}) &= (\mathbf{q} \cdot \bar{\mathbf{q}})^{\frac{1}{2}} \overbrace{(\mathbf{q} \cdot \mathbf{q})^{\frac{1}{2}}}^{> (\mathbf{q} \cdot \bar{\mathbf{q}})^{\frac{1}{2}}} - (\mathbf{q} \cdot \bar{\mathbf{q}}) > 0 \\ a + b &= \underbrace{\left(\frac{\mathbf{q} \cdot \bar{\mathbf{q}}}{\mathbf{q} \cdot \mathbf{q}} \right)^{\frac{1}{2}}}_{< 1} - 1 < 0. \end{aligned}$$

Hence, whenever $\mathbf{q} \neq \bar{\mathbf{q}}$ we can find a minimal and admissible bet as required. $\square$

In the context of prediction rules, one can see the above result as a failing of non-implicitly Bayesian ones; such prediction rules would induce non-exchangeable distributions over finite sequences of observations, and hence an agent that derives their beliefs from such a prediction rule would be vulnerable to the adversarial construction in Theorem 49.2.

49.4 Measuring Implicit Bayesianness

In the previous sections, we discussed what characterises a strictly implicitly-Bayesian prediction rule, how to turn a finite-horizon non-Bayesian rule into a Bayesian one, and what one leaves on the table by not doing so. Many believe that some ways of using deep learning, such as in-context learning, are already approximately Bayesian [1786]; see also Chapter 43. How could one verify whether this is the case? And how could one quantify how close it is to being true? We briefly digress in this section, to summarise how the findings from previous sections could be used towards the goal of *measuring* implicit Bayesianness.

The KL-gap The results in Propositions 49.1 and 49.2 directly suggest a way to quantify deviation from exchangeability (and hence implicit Bayesianness) at a fixed horizon by measuring how much performance is left on the table by not using the finitely exchangeable projection. For instance, Proposition 49.1 says that, for any full-support i.i.d. true data-generating distribution, we could ‘gain’ $D_{\text{KL}}(s_{1:n} \parallel \hat{s}_{1:n})$ in performance — as measured by reverse-KL proximity to p — by turning our finite-horizon prediction rule into an exchangeable one. Hence, this performance gap $D_{\text{KL}}(s_{1:n} \parallel \hat{s}_{1:n})$ could be used as a measure of deviation from exchangeability at horizon n . The gap is agnostic to the choice of exchangeable, full-support p . A prediction rule is only strictly exchangeable if the gap is 0 for all n . Nonetheless, a practitioner could pick some large n , and the metric is otherwise readily operationalisable.

The exchangeability gap In previous work, Mlodozeniec et al. [1192] have proposed directly measuring the average variation in the joint distribution implied by a prediction rule as the order of the elements in the sequence is varied. Of course, for an exchangeable prediction rule, the variation should be 0. Concretely, Mlodozeniec et al. [1192] suggested measuring the *variance* of the log-joint as they randomly sample permutations of the data uniformly at random. Denoting with Π a random variable that takes values in the set of all permutations of $\{1, \dots, n\}$ with equal probability, the measure was:

$$\mathbb{V} \left[\log \prod_{i=0}^{n-1} s_i(x_{\Pi(i+1)} | x_{\Pi(1)}, \dots, x_{\Pi(i)}) \right] = \mathbb{V} \left[\sum_{i=0}^{n-1} \log s_i(x_{\Pi(i+1)} | x_{\Pi(1)}, \dots, x_{\Pi(i)}) \right], \quad (49.15)$$

for a given sequence of datapoints $(x_1, \dots, x_n)$. The metric would then be averaged over different choices of $x_1, \dots, x_n \in \mathcal{X}^n$.

Arguably, the KL-gap metric has a stronger motivation, as it can be directly connected to desirable properties that being implicitly Bayesian brings. Nonetheless, Mlodozeniec et al. [1192] operationalised and computed the exchangeability metric for several practical algorithms, illustrating some counter-intuitive results, which we recapitulate and summarise in Figure 49.1.

Figure 49.1 shows empirically evaluated measures of implicit Bayesianness from [1192] for several Bayesian and non-Bayesian methods. They include a ‘strong’, but expensive, BNN baseline using Hamiltonian Monte Carlo inference (“*BNN MLP (HMC)*” in the plot), as well as a much more scalable but cruder Laplace approximation (“*BNN MLP (Laplace approx.)*” in the plot). These are compared with *exact* Gaussian Processes, as well as Gaussian processes where kernel hyperparameters are updated by maximising marginal likelihood. The latter is arguably much closer to how Gaussian Processes are used in practice. The plot also includes a regular MLP trained with SGD — with the same architecture as that used for the BNN methods — for reference.

Interestingly, the plot shows that the Laplace approximation for BNNs was *more* implicitly Bayesian than exact Gaussian process inference when the hyperparameters were fitted by maximising the marginal likelihood, following common practice. This shows that how strong of a Bayesian motivation an algorithm might *seem* to have might not necessarily correlate with how close to behaving like a Bayesian method it actually is.

49.5 Conclusion

In conclusion, this chapter summarises a different perspective on integrating Bayesian principles into deep learning, shifting the focus from explicit model specification to incorporating Bayesian principles at a *behavioural* level. We have shown what characterises Bayesian predictions, argued why Bayesian predictions are desirable in the first place, and have illustrated avenues for making an existing predictive algorithm Bayesian without engaging in explicit prior and likelihood construction.

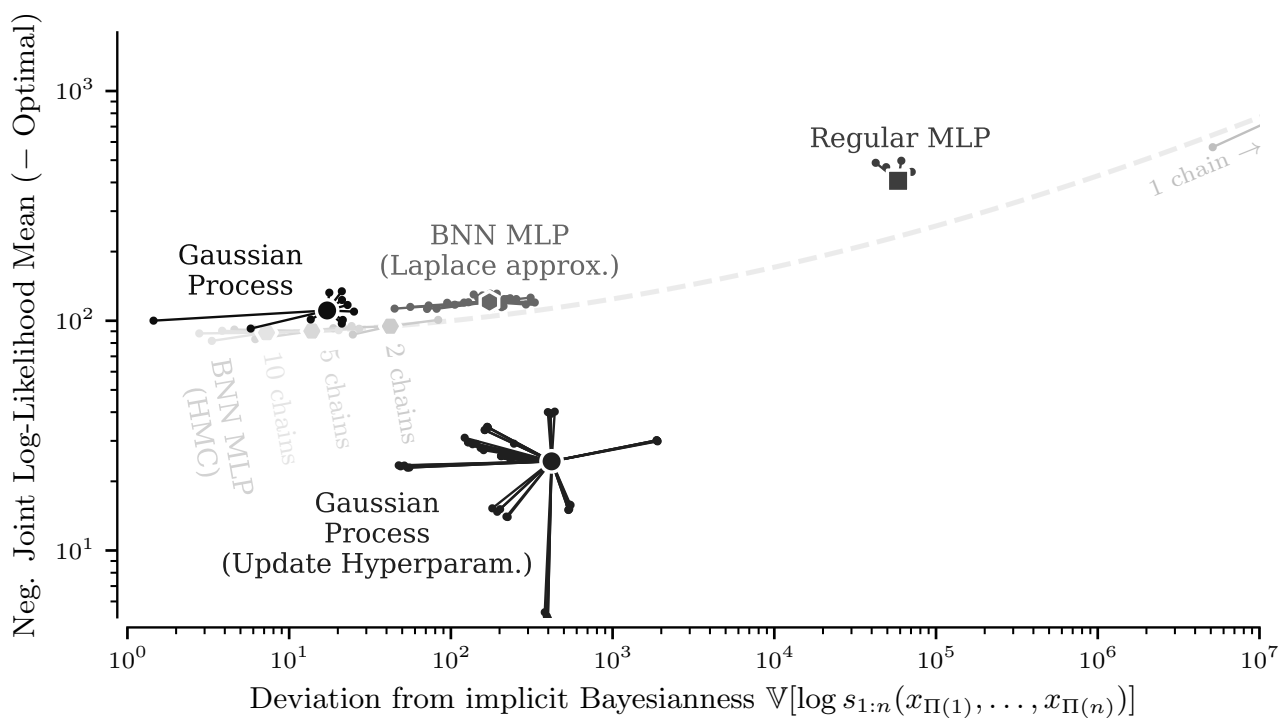


Figure 49.1: Empirically measuring implicit Bayesianness (x -axis) together with log-likelihood performance (y -axis) for several Bayesian Neural Network, Gaussian Process, and regular deep learning training configurations. Small dots indicate the metrics computed for an individual sequence of points, with a big dot at their centre indicating the average over different sequences of datapoints drawn from the regression task in [1192].

Chapter 50

Diffusion models

50.1 Introduction

Generative modelling, a cornerstone of modern machine learning, seeks to build models that can capture and sample from complex data distributions $p(\mathbf{x})$, such as those over natural images, audio waveforms, or molecular structures. In recent years, a new family of models, known as denoising diffusion models, has emerged, demonstrating an ability to generate high-fidelity, diverse samples that surpass the quality of other established model classes.

50.1.1 Defining Diffusion Models

A diffusion model is a latent variable model that generates samples from the data distribution $p(\mathbf{x})$ by iteratively removing noise from a sample generated from a tractable prior distribution. To train such a denoiser, one needs to obtain noisy samples with varying degrees of noise. From the perspective of non-equilibrium thermodynamics [1532], the noisy samples can be viewed as samples from an evolving system that moves from a low-entropy state towards a high-entropy state, for example, the diffusion of a gas in a container starting from some initial state. Concretely, a diffusion model consists of two discrete-time processes: a fixed forward noising process and a learned denoising process. The noising process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is Markovian and fixed, that is, it has no learnable parameters. It iteratively adds noise to a sample $\mathbf{x}_0 \sim q_{\text{data}}(\mathbf{x})$ over a sequence of timesteps $t = 1, \dots, T$. The noising process is defined such that its terminal distribution, $q(\mathbf{x}_T)$, converges to a simple, tractable prior distribution, such as an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The forward and reverse processes are illustrated in Figure 50.1 for a single data point $\mathbf{x}_0$.

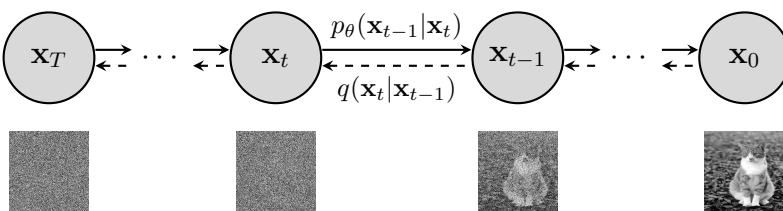


Figure 50.1: The forward process q gradually adds noise to the data, while the reverse process p_θ learns to denoise. Images show the progressive noising from $\mathbf{x}_0$ (clean) to $\mathbf{x}_T$ (pure Gaussian noise).

The denoising process $p_\theta(\mathbf{x}_{0:T})$ is a Markovian, parameterised process that iteratively removes noise from a sample $\mathbf{x}_T \sim q(\mathbf{x}_T)$ over a sequence of timesteps $t = T, \dots, 1$. It is parameterised by a neural network θ . Starting from a sample $\mathbf{x}_T \sim q(\mathbf{x}_T)$, the denoising process learns to perform a sequence of small *denoising* steps $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, such that the terminal distribution $p_\theta(\mathbf{x}_0)$ resembles the data distribution $q_{\text{data}}(\mathbf{x})$. To this end, the denoiser is trained to minimise the KL divergence between the denoised distribution and the data distribution, that is, $\min_\theta \text{D}_{\text{KL}}(p_\theta(\mathbf{x}_0)||q_{\text{data}}(\mathbf{x}))$.

50.1.2 Context within Probabilistic Generative Models

The rise of diffusion models can be understood by comparing them to the other families of generative models, each of which embodies a different set of trade-offs.

- **Variational Autoencoders (VAEs):** VAEs, like diffusion models, are latent variable models trained to optimise the Evidence Lower Bound (ELBO), a lower bound on the data log-likelihood $\log p(\mathbf{x})$. They consist of an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ that maps data to a latent space and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$ that reconstructs the data. VAEs provide a principled latent-variable framework with tractable variational training, but samples from standard VAE decoders are often reported to be overly smooth or of lower perceptual fidelity, especially when trained with pixelwise reconstruction objectives [892, 976].
- **Generative Adversarial Networks (GANs):** GANs reframe generative modelling as a two-player game between a generator $G_\theta(\mathbf{z})$ and a discriminator $D_\phi(\mathbf{x})$. The generator learns to transform prior noise $\mathbf{z}$ into samples, while the discriminator learns to distinguish the real data samples from the generated samples. At equilibrium, the generator produces samples indistinguishable from the true data. GANs produce sharper, higher-fidelity samples compared to VAEs. However, the adversarial objective is notoriously difficult to optimise [264], and these GANs suffer from mode collapse, where the generator learns to produce only a few high-quality samples. Furthermore, they lack a direct mechanism for estimating the likelihood of a data point.
- **Normalising Flows:** Discrete-time Normalising Flows model the data distribution directly by constructing an invertible, differentiable transformation $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$ that maps data $\mathbf{x}$ to a simple latent distribution $p_Z(\mathbf{z})$ (e.g., a Gaussian). The data likelihood can then be computed exactly using the change of variables formula: $\log p_X(\mathbf{x}) = \log p_Z(f(\mathbf{x})) + \log |\det(\mathbf{J}_f(\mathbf{x}))|$, where $\mathbf{J}_f$ is the Jacobian of f . This enables flows to be trained directly via maximum likelihood. While discrete-time normalising flows offer tractable likelihoods and exact latent inference, the need for an explicitly invertible f_θ with a tractable Jacobian determinant imposes significant architectural constraints. Continuous-time normalising flows relax this requirement by defining the map as the solution of an ODE, so that the network only models the vector field, and the log-density is then obtained from the instantaneous change of variables formula [272]. This continuous-time view is closely related to the probability flow ODE derived later in this chapter (Section 50.5.1), and to flow matching, which trains a neural vector field to transport noise to data without simulating an ODE during training [1054].

Diffusion models present a compelling alternative to the trilemma of sample quality, training stability, and likelihood estimation. They are trained with a stable objective similar to VAEs, which can be simplified to a simple regression loss. They produce state-of-the-art sample quality that rivals or exceeds that of GANs. Their primary drawback has been their slow sampling speed: generating a single sample requires T sequential evaluations of the neural network (where T is on the order of 1,000), making them orders of magnitude slower than single-shot models like VAEs or GANs. However, as we will discuss in Section 50.6.1, this limitation has been largely mitigated by recent algorithmic advancements.

50.1.3 Chapter Objectives and Roadmap

This chapter will provide a comprehensive, step-by-step derivation and analysis of denoising diffusion models.

1. We will begin by formalising the mathematical formulation of the fixed **forward perturbation process** (Section 50.2) and the parameterised **reverse generative process** in discrete time (Section 50.3).
2. We will then derive the **variational objective (ELBO)** for this model class (Section 50.4). This section contains the central derivation of the chapter, showing how the complex ELBO simplifies to a tractable, weighted regression objective. This is the key insight that makes diffusion models practical.
3. With the discrete-time model established, we will connect this formulation to its continuous-time limit (Section 50.5), where we will see that both the forward process and the reverse process can be seen as solutions to specific **Stochastic Differential Equations (SDEs)**. We will see how the coefficients of the SDE for the reverse process can be estimated using **denoising score matching**.
4. Finally, we will review the critical practical advancements that have made diffusion models the state-of-the-art, including **accelerated sampling algorithms (DDIM)**, methods for **conditional generation (guidance)**, and extensions to **discrete data domains** (Section 50.6 and Section 50.7).

50.2 The Forward Process: The Perturbation Kernel

The forward or the noising process represents the “*diffusion*” in diffusion models. This process is not learned; it is a fixed, predefined iterative mechanism for destroying the structure in a data point $\mathbf{x}_0$ and transforming it into pure noise.

50.2.1 A Fixed Markovian Process

The forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is defined as a finite horizon discrete-time Markov chain of length T . At each timestep $t \in [1, T]$, a small amount of Gaussian noise is added to the sample from the previous timestep, $\mathbf{x}_{t-1}$, which leads to the following transition kernel:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}). \quad (50.1)$$

Here, $\{\beta_t\}_{t=1}^T$ is a sequence of hyperparameters known as the variance schedule. These values are typically small and pre-determined. For example, a common choice is a linear schedule where β_t increases from $\beta_1 \approx 10^{-4}$ to $\beta_T \approx 0.02$. The schedule is designed such that $\beta_1 < \beta_2 < \dots < \beta_T$, meaning we add progressively more noise at later timesteps. The term $\sqrt{1 - \beta_t}$ is a scaling factor that keeps the variance of the signal from exploding. The entire forward process is a chain of these transitions, starting from the initial data $\mathbf{x}_0$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}).$$

50.2.2 The Marginal Distribution

A critical property of this noising process is that we can sample $\mathbf{x}_t$ at any arbitrary timestep t directly from $\mathbf{x}_0$ in a single step, without having to iterate through all T transitions. This is essential for making training tractable.

Let’s derive this closed-form marginal distribution $q(\mathbf{x}_t|\mathbf{x}_0)$. We can express $\mathbf{x}_{t-1}$ in terms of $\mathbf{x}_{t-2}$ and noise, and so on. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. As done in Chapter 13, we will use the reparameterisation trick [892], where sampling from $\mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$ is equivalent to computing $\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note, however, that since the forward process q is fixed, no gradients need to flow through this sampling, and the trick here merely yields a closed form for the marginal.

From Equation 50.1, we have:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t, \quad \text{where } \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (50.2)$$

Now, let’s recursively expand $\mathbf{x}_{t-1}$:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t} \left(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}}\boldsymbol{\epsilon}_{t-1} \right) + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1 - \alpha_{t-1})}\boldsymbol{\epsilon}_{t-1} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t \end{aligned}$$

where $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that the sum of two independent Gaussian variables $\mathcal{N}(0, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(0, \sigma_2^2\mathbf{I})$ is also a Gaussian $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$. The combined noise term satisfies

$$\sqrt{\alpha_t(1 - \alpha_{t-1})}\boldsymbol{\epsilon}_{t-1} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t \stackrel{d}{=} \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}, \quad \text{where } \bar{\boldsymbol{\epsilon}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

since the variance of the left-hand side is

$$(\alpha_t(1 - \alpha_{t-1}))\mathbf{I} + (1 - \alpha_t)\mathbf{I} = (\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t)\mathbf{I} = (1 - \alpha_t\alpha_{t-1})\mathbf{I}.$$

Here, $\bar{\boldsymbol{\epsilon}}$ represents a fresh standard Gaussian with the same distribution as the normalised linear combination. Thus

$$\mathbf{x}_t = \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}.$$

By induction, if we continue this expansion all the way back to $\mathbf{x}_0$, we get:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (50.3)$$

This is a key equation. It provides the marginal distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ in a simple, closed form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad (50.4)$$

This property is the workhorse of the training algorithm. It allows us to sample a $(\mathbf{x}_0, \mathbf{x}_t)$ pair for any t in a single step, without simulating the full chain.

By construction, the variance schedule $\{\beta_t\}$ is designed so that as $t \rightarrow T$, $\bar{\alpha}_T = \prod_{i=1}^T (1 - \beta_i) \rightarrow 0$. In this limit, Equation 50.4 becomes:

$$q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

Thus our choice of the forward process successfully transforms any data point $\mathbf{x}_0$ into a sample from the standard Gaussian prior distribution.

50.3 The Reverse Process: The Generative Kernel

Now that we have a process to destroy data, we need to learn a process to create it. This is the reverse process, $p_\theta(\mathbf{x}_{0:T})$, which learns to undo the forward diffusion, one step at a time.

50.3.1 A Parameterised Markovian Process

The reverse process is a Markov chain of length T that starts from the prior $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and generates a sequence $\mathbf{x}_{T-1}, \mathbf{x}_{T-2}, \dots, \mathbf{x}_0$.

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad (50.5)$$

The key question is how to learn the parameters of the reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. One obvious approach would be to try to match the true reverse transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as closely as possible. However, the true reverse transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is intractable, as it would require marginalizing over all possible $\mathbf{x}_0$: $q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \int q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_0|\mathbf{x}_t)d\mathbf{x}_0$.

The key to make the objective tractable, is to use the knowledge of $\mathbf{x}_0$. That is, if we know the original data point $\mathbf{x}_0$ that $\mathbf{x}_t$ was noised from, the reverse transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ would be tractable, which we can use to train the reverse process. Next, we analyse the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ to get a closed-form expression for it.

50.3.2 True Reverse Posterior

Let's analyse the true reverse posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. We can apply Bayes' theorem and the fact that the noising process is Markovian, i.e., $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1})$, to get:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_0|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t, \mathbf{x}_0)} \\ &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}. \end{aligned} \quad (50.6)$$

Plugging in the known Gaussian distributions for each term:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_{t-1}\mathbf{x}_0 + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C\right)\right) \end{aligned}$$

where C is a constant with respect to $\mathbf{x}_{t-1}$. This is a Gaussian distribution $\mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t, \tilde{\beta}_t\mathbf{I})$. By completing the square for $\mathbf{x}_{t-1}$, we can identify its variance $\tilde{\beta}_t$ and mean $\tilde{\boldsymbol{\mu}}_t$.

The precision (inverse variance) is:

$$\frac{1}{\tilde{\beta}_t} = \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} = \frac{\alpha_t(1 - \bar{\alpha}_{t-1}) + \beta_t}{\beta_t(1 - \bar{\alpha}_{t-1})} = \frac{1 - \bar{\alpha}_{t-1} - \beta_t + \bar{\alpha}_{t-1}\beta_t + \beta_t}{\beta_t(1 - \bar{\alpha}_{t-1})} = \frac{1 - \bar{\alpha}_t}{\beta_t(1 - \bar{\alpha}_{t-1})}$$

Therefore, the posterior variance is:

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \quad (50.7)$$

This variance is a deterministic function of the schedule hyperparameters and does not depend on the data.

The mean $\tilde{\mu}_t$ is given by:

$$\begin{aligned} \tilde{\mu}_t &= \left(\frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \tilde{\beta}_t / 2 \\ &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \\ &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 \end{aligned} \quad (50.8)$$

So, the true reverse posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is Gaussian with a known mean and variance:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad (50.9)$$

50.3.3 Parameterising the Generative Transition

The model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ for the reverse process is also parameterised as a Gaussian:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \quad (50.10)$$

Since the posterior is also Gaussian, this choice will greatly simplify the training objective, as we will see in the next section. In Section 50.5.1, we will use the theory of stochastic differential equations to see that for large T , the reverse process is also Gaussian [33, 1541], providing further justification for this parameterisation.

50.4 Optimisation Objective: From an Evidence Lower Bound to a Simplified Denoising Loss

With the forward and reverse processes defined, we can now derive the objective function used to train the model. The model is trained, like a VAE (Section 13.3.3), by maximising the Evidence Lower Bound (ELBO) on the data log-likelihood $\log p_\theta(\mathbf{x}_0)$ under the denoising distribution, but with a fixed variation distribution, i.e. the noising process. See Section 12.1.3 of Chapter 12 for understanding the role of ELBO in variational inference.

50.4.1 The Variational Lower Bound (ELBO)

We wish to maximise $\mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}}[\log p_\theta(\mathbf{x}_0)]$. We can write:

$$\begin{aligned} \log p_\theta(\mathbf{x}_0) &= \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\ &= \log \int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \\ &= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\ &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \quad (\text{Jensen's Inequality}) \end{aligned}$$

This last term is the ELBO, which, with a minus sign, can be used as the training objective L_{VLB} . Note that the Jensen's-inequality argument used here is the same one used to derive the ELBO on the model evidence in

Section 12.1.5. We can decompose this into a more interpretable form:

$$\begin{aligned}
L_{\text{VLB}}(\mathbf{x}_0, \theta) &= -\mathbb{E}_q \left[\log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
&= -\mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t=1}^T \log p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) - \sum_{t=1}^T \log q(\mathbf{x}_t|\mathbf{x}_{t-1}) \right] \\
&= -\mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t=2}^T \log p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right. \\
&\quad \left. - \log q(\mathbf{x}_1|\mathbf{x}_0) - \sum_{t=2}^T \log q(\mathbf{x}_t|\mathbf{x}_{t-1}) \right]
\end{aligned}$$

Using the identity $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}$ (from the Markov property and Bayes' rule), we can rewrite $\log q(\mathbf{x}_t|\mathbf{x}_{t-1})$ as $\log q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) + \log q(\mathbf{x}_t|\mathbf{x}_0) - \log q(\mathbf{x}_{t-1}|\mathbf{x}_0)$. Substituting this in:

$$\begin{aligned}
L_{\text{VLB}} &= -\mathbb{E}_q \left[\log p(\mathbf{x}_T) - \log q(\mathbf{x}_T|\mathbf{x}_0) + \sum_{t=2}^T (\log p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) - \log q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)) \right. \\
&\quad \left. - \log q(\mathbf{x}_1|\mathbf{x}_0) + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \sum_{t=2}^T (\log q(\mathbf{x}_{t-1}|\mathbf{x}_0) - \log q(\mathbf{x}_t|\mathbf{x}_0)) \right]
\end{aligned}$$

The last summation is a telescoping sum $\log q(\mathbf{x}_1|\mathbf{x}_0) - \log q(\mathbf{x}_T|\mathbf{x}_0)$, which cancels terms.

$$\begin{aligned}
L_{\text{VLB}} &= -\mathbb{E}_q \left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^T \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \\
&= \mathbb{E}_q \left[\underbrace{\text{D}_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \underbrace{\sum_{t=2}^T \text{D}_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{1:T-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right] \quad (50.11)
\end{aligned}$$

This gives us a sum of KL divergences and a reconstruction term:

- $L_T = \text{D}_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))$: The **prior matching term**. This measures how well the final noised distribution $q(\mathbf{x}_T|\mathbf{x}_0)$ matches the prior $p(\mathbf{x}_T)$. Since q is fixed and p is a fixed $\mathcal{N}(\mathbf{0}, \mathbf{I})$, this term is a constant and can be ignored during optimisation.
- $L_{t-1} = \text{D}_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$: The **denoising matching terms** (each term L_{t-1} , summed as $L_{1:T-1}$ in Eq. (50.11)). This is a sum from $t = 2$ to T . Each term compares the learned reverse transition p_θ to the true (but $\mathbf{x}_0$ -conditional) reverse posterior q . This is the primary learning signal.
- $L_0 = \mathbb{E}_q[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]$: The **reconstruction term**. This term measures how well the model reconstructs the final data $\mathbf{x}_0$ from the last latent state $\mathbf{x}_1$.

Following Ho et al. [755], DDPM fixes the reverse-process variance to the schedule value $\tilde{\beta}_t \mathbf{I}$ rather than learning the general covariance $\Sigma_\theta(\mathbf{x}_t, t)$ introduced in Equation 50.10. The optimisation objective is to minimise the loss $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim q_{\text{data}}}[L_{\text{VLB}}(\mathbf{x}_0, \theta)]$.

50.4.2 Simplification via Reparameterisation

Naive Monte Carlo estimation of Equation 50.11 will have high variance. We can use the fact that both

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \\
p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I}),
\end{aligned}$$

are Gaussian and compute the KL divergence in closed form. When the variances of the two Gaussians are identical ($\sigma_1^2 = \sigma_2^2 = \tilde{\beta}_t$), the KL divergence simplifies to a simple L_2 distance between their means, scaled by the variance:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right]. \quad (50.12)$$

The training objective is therefore to train a neural network $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ to predict the true posterior mean $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$. As noted in Ho et al. [755] that this objective can be simplified even further by writing $\tilde{\boldsymbol{\mu}}_t$ purely in terms of $\mathbf{x}_t$ and the noise $\boldsymbol{\epsilon}$ using the forward process relationship from Equation 50.3. Substituting

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}) \quad (50.13)$$

in the expression for $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ from Equation 50.8, we get:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right). \quad (50.14)$$

This motivates a new parameterisation for our model $\boldsymbol{\mu}_\theta$. Instead of training a network $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t$ directly, let's train a network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ to predict the noise $\boldsymbol{\epsilon}$. We can then express our predicted mean $\boldsymbol{\mu}_\theta$ using the same functional form as $\tilde{\boldsymbol{\mu}}_t$:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \quad (50.15)$$

50.4.3 The Simplified Loss Function

We now substitute the $\boldsymbol{\epsilon}$ -prediction parameterisation back into our L_{t-1} loss term:

$$\begin{aligned} L_{t-1} &= \mathbb{E}_q \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\boldsymbol{\mu}}_t - \boldsymbol{\mu}_\theta\|^2 \right] \\ &= \mathbb{E}_q \left[\frac{1}{2\tilde{\beta}_t} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right) - \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \right\|^2 \right] \\ &= \mathbb{E}_q \left[\frac{1}{2\tilde{\beta}_t} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \right) (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}) \right\|^2 \right] \\ &= \mathbb{E}_q \left[\frac{1}{2\tilde{\beta}_t} \frac{1}{\alpha_t} \frac{\beta_t^2}{1 - \bar{\alpha}_t} \|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|^2 \right] \\ &= \mathbb{E}_{x_0, \boldsymbol{\epsilon}, t} \left[\left(\frac{\beta_t^2}{2\tilde{\beta}_t \alpha_t (1 - \bar{\alpha}_t)} \right) \|\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}\|^2 \right] \end{aligned} \quad (50.16)$$

This is the principal objective function. The task of minimising the ELBO (a sum of KL divergences) has been reduced to a simple, weighted L_2 regression problem. We just need to train a neural network $\boldsymbol{\epsilon}_\theta$ (typically a U-Net, shown in Figure 50.2) to predict the noise $\boldsymbol{\epsilon}$ that was added to $\mathbf{x}_0$ to create $\mathbf{x}_t$.

Ho et al. [755] noted that training was more stable and effective when ignoring the complex weighting term in front of the L_2 loss. They propose a simplified objective:

$$L_{\text{simple}}(\theta) = \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} [\|\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}\|^2] \quad (50.17)$$

This is the final, practical objective function used by most modern diffusion models. For training, the L_0 reconstruction term can also be shown to be equivalent to this form for $t = 1$, so this single loss function L_{simple} is all that is needed.

50.4.4 Training and Sampling Algorithms

The simplified objective in Equation 50.17 leads to a very simple training algorithm.

Algorithm 36: DDPM Training**Data:** Data distribution $\mathbf{x}_0 \sim q_{\text{data}}(\mathbf{x}_0)$, timesteps T , model ϵ_θ **repeat** Sample $\mathbf{x}_0 \sim q_{\text{data}}(\mathbf{x}_0)$ Sample $t \sim \text{Uniform}(\{1, \dots, T\})$ Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ Compute $\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon$

Take gradient descent step on:

$$\nabla_\theta \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2$$

until *convergence*

The sampling algorithm (generation) works by running the reverse process from $t = T$ down to $t = 1$. At each step, we use our network ϵ_θ to predict the noise, and then use that to compute the mean of the distribution for $\mathbf{x}_{t-1}$.

Algorithm 37: DDPM Ancestral Sampling**Result:** A sample $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0)$ Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ **for** $t = T, \dots, 1$ **do** Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$ Compute the predicted noise $\epsilon_\theta = \epsilon_\theta(\mathbf{x}_t, t)$ Compute the predicted mean μ_θ using Eq. 50.15:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta \right)$$

Compute the sample for the previous timestep:

$$\mathbf{x}_{t-1} = \mu_\theta + \sqrt{\tilde{\beta}_t} \mathbf{z}$$

end**return** $\mathbf{x}_0$

50.5 Continuous-Time Diffusion: Score-Based Generative Models

The discrete-time formulation discussed so far provides a practical algorithm, but it leaves open some deeper questions. For example, why does a fixed variational distribution q , which is a Gaussian, work at all? What are some other design choices that can improve the performance of the model? A deeper mathematical understanding of diffusion models can be found by analysing its continuous-time limit as $T \rightarrow \infty$. This connects the discrete-time formulation to score-based generative modelling [1540, 1541] and ultimately to the rich field of SDEs.

50.5.1 Convergence to a Stochastic Differential Equation (SDE)

As the number of timesteps $T \rightarrow \infty$ and the per-step perturbation $\beta_t \rightarrow 0$, the discrete forward process (Equation 50.1) converges to the solution of a continuous-time Itô SDE:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t) dt + g(t) d\mathbf{w}, \quad (50.18)$$

which describes the evolution of a variable $\mathbf{x}(t)$ (which we will write as $\mathbf{x}_t$) subject to a deterministic drift function $\mathbf{f}(\mathbf{x}_t, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ and a stochastic diffusion coefficient $g(t) : \mathbb{R} \rightarrow \mathbb{R}$. Here, t is now a continuous variable, typically normalised to $t \in [0, 1]$ or $[0, T]$, and $\mathbf{w}$ is the standard Wiener process.

Different choices of discrete processes converge to different SDEs. The framework introduced by Song et al. [1541] unifies two major lines of work:

1. **Variance-Preserving (VP) SDE:** This SDE corresponds to the continuous limit of the DDPM formulation [755]. To derive this rigorously, we start with the discrete Markov chain from Eq. 50.2:

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \mathbf{z}_{t-1}, \quad t = 1, \dots, N \quad (50.19)$$

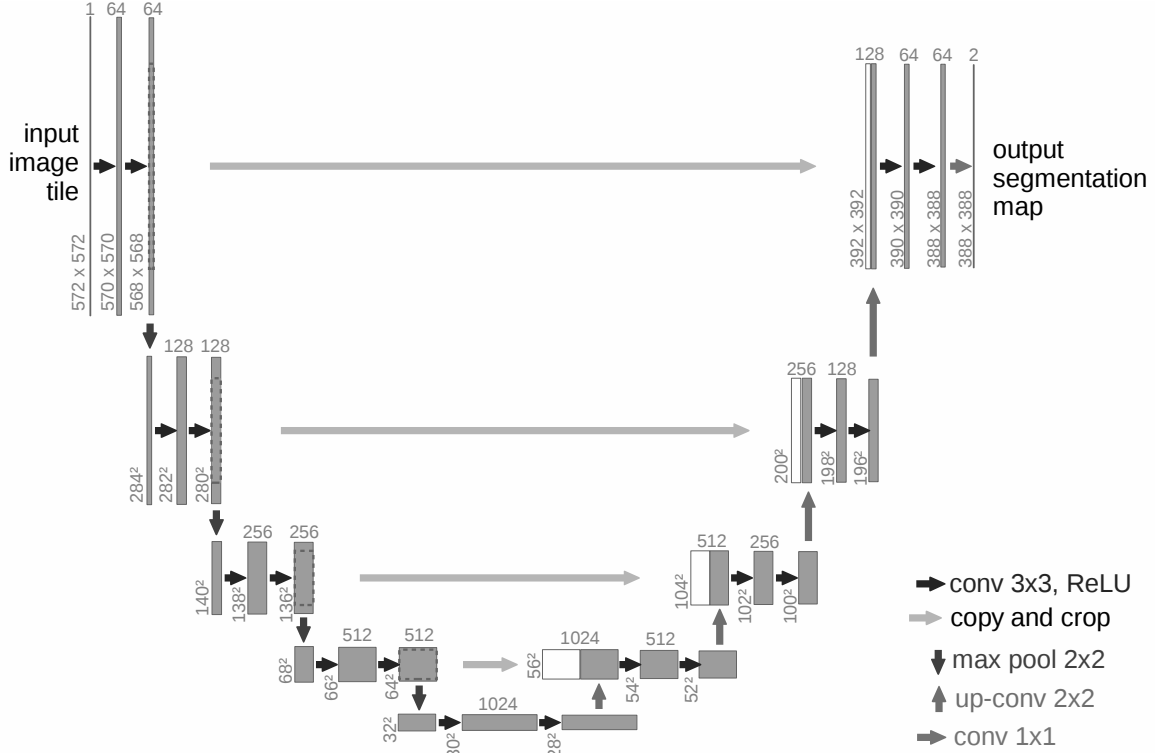


Figure 50.2: A diagram illustrating the U-Net architecture commonly used for the noise-prediction network ϵ_θ . The skip connections are vital for preserving high-resolution details from $\mathbf{x}_t$ while the network processes information at coarser scales. The figure is adapted from Ronneberger et al. [1419].

where $\mathbf{z}_{t-1} \sim \mathcal{N}(0, \mathbf{I})$. To take the continuum limit as $N \rightarrow \infty$, we introduce an auxiliary set of noise scales $\{\bar{\beta}_t = N\beta_t\}_{t=1}^N$ and rewrite the discrete update as [1540]:

$$\mathbf{x}_t = \sqrt{1 - \frac{\bar{\beta}_t}{N}} \mathbf{x}_{t-1} + \sqrt{\frac{\bar{\beta}_t}{N}} \mathbf{z}_{t-1}. \quad (50.20)$$

In the limit of $N \rightarrow \infty$, the sequence $\{\bar{\beta}_t\}_{t=1}^N$ becomes a function $\beta(t)$ indexed by $t \in [0, 1]$, with $\beta(t/N) = \bar{\beta}_t$. Similarly, $\mathbf{x}_t$ becomes $x(t)$ and $\mathbf{z}_t$ becomes $z(t)$. With time step $\Delta t = 1/N$, we can write the discrete update for $t \in \{0, 1, \dots, (N-1)/N\}$ as:

$$\begin{aligned} \mathbf{x}(t + \Delta t) &= \sqrt{1 - \beta(t + \Delta t)\Delta t} \mathbf{x}(t) + \sqrt{\beta(t + \Delta t)\Delta t} \mathbf{z}(t) \\ &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t + \Delta t)\Delta t \mathbf{x}(t) + \sqrt{\beta(t + \Delta t)\Delta t} \mathbf{z}(t) \\ &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t \mathbf{x}(t) + \sqrt{\beta(t)\Delta t} \mathbf{z}(t), \end{aligned} \quad (50.21)$$

where the approximations hold when $\Delta t \ll 1$. Using the expansion $\sqrt{1 - u} \approx 1 - \frac{1}{2}u$ for small u , and assuming $\beta(t)$ varies slowly over Δt . In the limit $\Delta t \rightarrow 0$, the discrete differences $\mathbf{x}(t + \Delta t) - \mathbf{x}(t)$ converge to the differential $d\mathbf{x}(t)$, and $\mathbf{z}(t)\sqrt{\Delta t}$ converges to the Wiener process increment $d\mathbf{w}_t$. Thus, Eq. 50.21 yields the VP SDE:

$$d\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t) dt + \sqrt{\beta(t)} d\mathbf{w}_t. \quad (50.22)$$

This is called “variance-preserving” because as the process diffuses forward in time, the variance of $x(t)$ remains bounded and converges to 1 at $t = T$, matching the prior distribution.

2. **Variance-Exploding (VE) SDE:** This SDE corresponds to an earlier line of work on Noise-Conditioned Score Matching [1540], which used annealed Langevin dynamics. This process is defined by $d\mathbf{x}_t = \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\mathbf{w}_t$. Here, the drift is zero, and the process simply adds noise of increasing variance $\sigma^2(t)$, causing the variance of $p_t(\mathbf{x}_t)$ to grow monotonically with t .

This SDE framework is general and can encompass many different noising processes. The key is that they all transform a complex $p_0(\mathbf{x}_0)$ into a simple $p_T(\mathbf{x}_T)$.

50.5.2 The Reverse-Time SDE

The time-reversal of a stochastic process is a process that runs backwards in time. For the stochastic process governed by the SDE of the form in Equation 50.18, a key result from the theory of SDEs by Anderson [33] states that the time-reversal of this process is also governed by an SDE of the same form:

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)] dt + g(t) d\bar{\mathbf{w}}_t \quad (50.23)$$

where $d\bar{\mathbf{w}}_t$ is a reverse-time Wiener process, and dt is an infinitesimal time step in the reverse direction. This reverse-time SDE runs from $t = T$ to $t = 0$ and transforms samples from the prior p_T back into samples from the data distribution p_0 .

Comparing the forward SDE (Equation 50.18) and the reverse SDE (Equation 50.23), we see they are identical except for the drift term. The reverse drift is the forward drift *minus* a new term that depends on $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$. Figure 50.3 shows some trajectories of the forward and reverse SDEs for a simple 1D mixture of Gaussian data distribution.

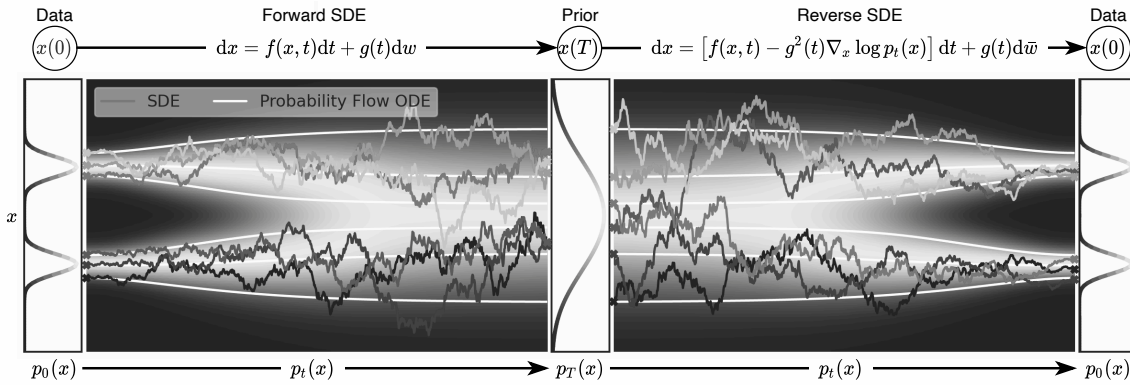


Figure 50.3: The forward and reverse SDEs for a 1D toy data distribution, which is a mixture of two Gaussians. On the left are some simulated trajectories of the forward SDE that converts samples from the data distribution p_0 to samples from the prior p_T . On the right are the trajectories of the reverse SDE with learnt score network that converts samples from the prior p_T back to samples from the data distribution p_0 . The figure is adapted from Song et al. [1541].

50.5.3 The Score Function: Unification of Objectives

The only unknown quantity in this reverse SDE is $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$. This vector field is known as the score function of the marginals $p_t(\mathbf{x}_t)$. It points in the direction of steepest ascent for the log-density at any point $\mathbf{x}_t$. This immediately reveals the fundamental task of score-based generative modelling: if we can train a neural network $\mathbf{s}_\theta(\mathbf{x}_t, t)$ to estimate the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$ for all $t \in [0, T]$, we can plug $\mathbf{s}_\theta$ into the reverse SDE and solve it numerically to generate new samples.

The training objective for this score network $\mathbf{s}_\theta$ is the score-matching objective, which minimises the Fisher divergence between the model and the true data distribution:

$$L(\theta) = \mathbb{E}_t \mathbb{E}_{p_t(\mathbf{x}_t)} [\|\mathbf{s}_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\|_2^2]$$

This objective is intractable as written, because it requires access to the score of the unknown distribution $p_t(\mathbf{x}_t)$. However, Vincent [1680] shows that this objective is equivalent to minimising:

$$L(\theta) = \mathbb{E}_t \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \mathbb{E}_{p_0(\mathbf{x}_0)} [\|\mathbf{s}_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)\|_2^2],$$

called the denoising score-matching objective. This new objective is tractable, because $q(\mathbf{x}_t|\mathbf{x}_0)$ is our known, analytic forward perturbation kernel (Eq. 50.4).

Let's compute this tractable score:

$$\begin{aligned}\log q(\mathbf{x}_t|\mathbf{x}_0) &= \log \left(C \cdot \exp \left(-\frac{1}{2(1-\bar{\alpha}_t)} \|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\|^2 \right) \right) \\ &= \log C - \frac{1}{2(1-\bar{\alpha}_t)} \|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\|^2 \\ \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0) &= -\frac{1}{1-\bar{\alpha}_t} (\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)\end{aligned}$$

Using our reparameterisation from Eq. 50.3, $\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0 = \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{1}{1-\bar{\alpha}_t} (\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}) = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}} \quad (50.24)$$

This is a key result. The true score of the perturbed data distribution is just the (scaled) negative noise that was added to obtain $\mathbf{x}_t$ from $\mathbf{x}_0$. If we parameterise our score network $\mathbf{s}_\theta$ in a way that maps to the ϵ -predictor, $\mathbf{s}_\theta(\mathbf{x}_t, t) = -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$, and plug this into the score-matching loss:

$$\begin{aligned}L(\theta) &\propto \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}} - \left(-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}} \right) \right\|_2^2 \right] \\ &= \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{1-\bar{\alpha}_t} \|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|_2^2 \right]\end{aligned}$$

This is precisely the weighted L_{simple} loss from Equation 50.17. Hence, we note the following equivalence: Minimising the DDPM's ELBO is equivalent to training a denoising score-matching model. The ϵ -predictor $\boldsymbol{\epsilon}_\theta$ is just a reparameterised version of the score function $\mathbf{s}_\theta$. Specifically, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx -\sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$. This unified view is powerful, as it allows us to decouple the training objective from the sampling method.

50.5.4 Sampling via SDE and ODE Solvers

With a trained score model $\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$, we can now plug it into the reverse SDE (Equation 50.23) and solve it numerically from $t = T$ to $t = 0$. This opens up a wide variety of solvers beyond the original DDPM ancestral sampler in Algorithm 37.

Numerical SDE Solvers

These methods discretise the reverse-time SDE and add the appropriate stochastic noise at each step.

- **Euler-Maruyama:** This is the simplest (and most common) SDE solver. We discretise time into N steps $T = t_N, t_{N-1}, \dots, t_0 = 0$, with $\Delta t < 0$ being the step size.

$$\mathbf{x}_{i-1} = \mathbf{x}_i + [\mathbf{f}(\mathbf{x}_i, t_i) - g(t_i)^2 \mathbf{s}_\theta(\mathbf{x}_i, t_i)] \Delta t + g(t_i) \sqrt{|\Delta t|} \mathbf{z}_i \quad (50.25)$$

where $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I})$. This is a general sampler that adds noise at each step. The DDPM ancestral sampler in Algorithm 37 can be viewed as a different discretisation of the same reverse-time SDE.

Predictor-Corrector Samplers

Due to discretisation error, a single step of an SDE solver (the predictor) may not land the sample $\mathbf{x}_{i-1}$ in a high-density region of $p_{t_{i-1}}$, and if unchecked these errors accumulate over steps. Since we have the learned score function at different noise levels, we can use it to “correct” the sample $\mathbf{x}_{i-1}$ by pushing it towards a higher-density region of $p_{t_{i-1}}$ using MCMC [1541].

- **Langevin Dynamics Corrector:** After the predictor step (Equation 50.25) gives an initial $\hat{\mathbf{x}}_{i-1}$, we can run K steps of MCMC using annealed Langevin dynamics, which targets $p_{t_{i-1}}$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\gamma}{2} \nabla_{\mathbf{x}} \log p_{t_{i-1}}(\mathbf{x}^{(k)}) + \sqrt{\gamma} \mathbf{z}_k$$

We just plug in our score model $\mathbf{s}_\theta(\mathbf{x}^{(k)}, t_{i-1})$ for the gradient. This predictor-corrector loop (SDE step followed by MCMC steps) is very robust and was the core of the original noise conditioned score network models [1540].

Probability Flow ODE Solvers

Remarkably, there exists a corresponding *deterministic* process, defined by an Ordinary Differential Equation (ODE), whose trajectories, when initialized using the prior distribution, evolve to generate paths with the marginal densities $p_t(\mathbf{x}_t)$ that match those of the paths from the SDE. This is the probability flow ODE:

$$d\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) \right] dt \quad (50.26)$$

This is a deterministic equation. If we solve it backwards in time from $t = T$ to $t = 0$ (using our score model $\mathbf{s}_\theta$ for the gradient $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$), we get a deterministic mapping from an $\mathbf{x}_T$ to some $\mathbf{x}_0$. We can solve the ODE using any black-box numerical ODE solver (e.g., Runge-Kutta, Heun’s method, etc.). Figure 50.3 shows the trajectories of the probability flow ODE alongside those from the SDE. The use of ODEs instead of SDEs offers several advantages:

1. The sampling path is deterministic, which allows inversions, that is, going from a data sample back to the latent space. ODEs also allow computing exact log-likelihood using the instantaneous change of variables formula [272].
2. We can use high-order, adaptive-step-size ODE solvers to get a more accurate solution with fewer network evaluations.
3. The equivalence provides the theoretical foundation for the DDIM sampler (Section 50.6), which can be shown to be a first-order discretisation of this probability flow ODE.

Although theoretically, the reverse SDE and the probability flow ODE share the same marginal densities p_t in continuous time, the practical choice between them depends on the computational budget and on the desired properties of the sampler. ODE solvers have lower discretisation error and therefore are preferable when few network evaluations (NFEs) are available. However, in the high-NFE regime, ODE solvers plateau, while SDE solvers achieve better sample quality as the injected stochasticity contracts accumulated score-estimation and discretisation errors along the trajectory [1541]. Hybrid samplers that introduce controlled stochasticity can achieve high-quality samples at lower NFEs than pure SDE solvers [873, 1790].

50.6 Improving Sampling Efficiency and Controllability

While the core DDPM/SDE framework is conceptually powerful, its practical utility has been unlocked by several key algorithmic improvements. The most important address two major limitations: the slow, iterative sampling process and the lack of a built-in mechanism for conditional generation.

50.6.1 Accelerated Sampling: Denoising Diffusion Implicit Models (DDIM)

The DDPM ancestral sampler (Algorithm 37) is slow, requiring T (e.g., 1000) sequential network evaluations. This is because it is *stochastic*—at each step t , it samples from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by adding noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \tilde{\beta}_t \mathbf{I})$. This procedure correctly models the generative process p_θ , which is a reversal of the Markovian forward process q . However, Song et al. [1538] observed that the simplified DDPM loss L_{simple} (Equation 50.17) only depends on the marginal distributions $q(\mathbf{x}_t|\mathbf{x}_0)$, not the full joint distribution $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$. This means the network ϵ_θ is trained on $(\mathbf{x}_0, \mathbf{x}_t)$ pairs, regardless of *how* $\mathbf{x}_t$ was reached from $\mathbf{x}_0$.

DDIM [1538] exploits this observation by defining a *new*, non-Markovian forward process with marginals

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) := \mathcal{N} \left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}, \sigma_t^2 \mathbf{I} \right),$$

which has the same marginals $q(\mathbf{x}_t|\mathbf{x}_0)$ as the original DDPM process. This new process allows for a more general, deterministic reverse process. The DDIM update rule is:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \cdot \left(\frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \epsilon_t \quad (50.27)$$

Note the new parameter σ_t , which controls the stochasticity. If we set $\sigma_t \approx 0$ for all t , the process becomes deterministic. This means that for a given starting noise $\mathbf{x}_T$, the entire sampling trajectory $\mathbf{x}_{T-1}, \dots, \mathbf{x}_0$ is fixed.

This resulting model is called *implicit* because $\mathbf{x}_0$ is generated deterministically from $\mathbf{x}_T$. This deterministic update is similar to the discretisation of the probability flow ODE (Eq. 50.26).

Another benefit of this formulation is that the generative process is no longer tied to the original T noising steps used to train the model. We can define a new, shorter sampling trajectory using a subsequence of timesteps $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_S\}$ where $S \ll T$. We can then use the same network ϵ_θ (trained on all T steps) but only evaluate it at the timesteps in $\mathcal{T}$, jumping from $\mathbf{x}_{\tau_i}$ to $\mathbf{x}_{\tau_{i+1}}$. This enables a 10-50x acceleration in sampling speed (wall clock time) with a surprisingly small drop in sample quality.

50.6.2 Conditional Generation

The models discussed so far are unconditional: they sample from the entire data distribution $p(\mathbf{x})$. However, what if we want to generate data samples of a specific class? For instance, imagine we want an image generator to generate images of dogs. This problem can be framed as conditional generation, wherein we want to sample from $p(\mathbf{x}|y)$ instead of $p(\mathbf{x})$. Here, y is the conditioning variable.

Classifier Guidance

One way to generate samples from $p(\mathbf{x}|y)$ is to use the Bayes' rule on the score function:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|y) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(y|\mathbf{x}).$$

We already have a model for the unconditional score: $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \mathbf{s}_\theta(\mathbf{x}_t, t)$. The new term $\nabla_{\mathbf{x}} \log p_t(y|\mathbf{x})$ is the gradient of a classifier $p_\phi(y|\mathbf{x}_t)$ trained to predict the class y from the *noised* data $\mathbf{x}_t$. This approach, wherein we use a separate classifier to guide the sampling is referred to as *classifier guidance* [422].

Intuitively, we are steering the score-based sampler by plugging in a modified score $\hat{\mathbf{s}}_\theta$:

$$\hat{\mathbf{s}}_\theta(\mathbf{x}_t, t, y) = \mathbf{s}_\theta(\mathbf{x}_t, t) + w \cdot \nabla_{\mathbf{x}} \log p_\phi(y|\mathbf{x}_t),$$

where, w is the guidance scale, a scalar that controls the strength of the conditioning. In the ϵ -prediction formulation, this becomes:

$$\hat{\epsilon}_\theta(\mathbf{x}_t, t, y) = \epsilon_\theta(\mathbf{x}_t, t) - w \cdot \sqrt{1 - \bar{\alpha}_t} \cdot \nabla_{\mathbf{x}} \log p_\phi(y|\mathbf{x}_t) \quad (50.28)$$

This modified $\hat{\epsilon}_\theta$ is then plugged into the DDPM or DDIM sampling algorithm. This technique works quite well, trading off diversity for stronger adherence to the condition y . Moreover, since the classifier is a separate model, we are free to change it without re-training the generator.

50.6.3 Classifier-Free Guidance

Classifier guidance is effective, but it is computationally burdensome. It requires training a separate, noise-conditioned classifier $p_\phi(y|\mathbf{x}_t)$, which must be evaluated at every sampling step, adding to the cost.

Ho and Salimans [753] proposed a simpler and more powerful alternative: classifier-free guidance (CFG). The key idea is to train a *single* diffusion model $\epsilon_\theta(\mathbf{x}_t, t, y)$ that is jointly conditioned on the timestep t and the conditioning signal y (which can be a class label, a text embedding, etc.). During training, this conditioning y is randomly replaced with a suitable “null” token (e.g., 10-20% of the time), which implies no conditioning. This forces the same network to learn both the conditional model $\epsilon_\theta(\mathbf{x}_t, t, y)$ and the unconditional model $\epsilon_\theta(\mathbf{x}_t, t, \emptyset)$.

During sampling, we can then perform an extrapolation. The guided ϵ -prediction is computed as:

$$\hat{\epsilon}_\theta(\mathbf{x}_t, t, y) = (1 + w) \epsilon_\theta(\mathbf{x}_t, t, y) - w \epsilon_\theta(\mathbf{x}_t, t, \emptyset) \quad (50.29)$$

This simple modification is incredibly powerful. The term $\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, \emptyset)$ can be viewed as an estimate of the classifier gradient, learned by the network itself. Equivalently, with guidance scale defined as $s := w + 1$, Eq. (50.29) can be written as $\hat{\epsilon}_\theta(\mathbf{x}_t, t, y) = \epsilon_\theta(\mathbf{x}_t, t, \emptyset) + s(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, \emptyset))$, where $s = 1$ recovers the conditional prediction and $s = 0$ corresponds to unconditional generation. Typical values of the guidance scale s for strong conditional generation are around 7.5. Larger values can reduce sample diversity. CFG is the *de facto* standard for modern large-scale text conditional diffusion models.

50.6.4 Latent Diffusion Models

The improvements discussed so far reduce the number of sampling steps. Latent Diffusion Models (LDMs) [1417] provide a complementary approach that reduces the cost of each step. An LDM is trained in two stages. First, a regularized autoencoder, consisting of an encoder $\mathcal{E}$ and decoder $\mathcal{D}$, maps an image $\mathbf{x}$ to a spatially compressed latent representation

$$\mathbf{z} = \mathcal{E}(\mathbf{x}),$$

from which the image can be approximately reconstructed as

$$\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{z}).$$

The autoencoder is then frozen, and a diffusion model is trained on the distribution of encoded representations rather than directly on pixels.

At generation time, the learned reverse process produces a clean latent $\mathbf{z}_0$, which is decoded into an image as

$$\hat{\mathbf{x}} = \mathcal{D}(\mathbf{z}_0).$$

Because the autoencoder absorbs part of the burden of representing local pixel-level variation, the diffusion model can operate on a substantially smaller spatial input. For example, with an $8\times$ spatial downsampling factor, a 512×512 RGB image corresponds to a 64×64 latent grid, reducing the number of spatial locations processed by the diffusion model by a factor of 64.

This two-stage design resembles a latent-variable generative model in which the diffusion model serves as a learned prior over the representations produced by the encoder. Unlike a conventional VAE, however, the autoencoder and latent prior are trained sequentially rather than jointly through a single variational objective. Combined with cross-attention conditioning and classifier-free guidance, this design underpins many large-scale text-to-image systems.

50.7 Diffusion Models for Discrete Data Domains

The entire framework presented so far, from the Gaussian forward process to the L_2 denoising loss, has been designed for continuous data $\mathbf{x} \in \mathbb{R}^n$ (e.g., pixels, audio samples). For discrete, structured data (e.g., text, molecules, graphs), this framework is not directly applicable.

50.7.1 Beyond Gaussian Perturbations

Applying Gaussian noise to discrete data (e.g., one-hot vectors representing text tokens) immediately destroys the discrete structure, pushing the data off its original manifold into a continuous space. While some work [423, 1033] has explored “embedding” discrete data in a continuous space and diffusing there, a more direct approach is to define a diffusion-like process *natively* in the discrete state-space.

50.7.2 Discrete-State Markov Processes

Austin et al. [63], Hoogetboom et al. [780] formalise diffusion like noising process for discrete space using discrete-time Markov chains.

Forward Process

Assume that we are working with sequences of fixed length L , with each entry taking values in a finite vocabulary $\mathcal{V}$. Then the Gaussian kernel is replaced by a discrete-state transition kernel for the full sequence $\mathbf{x}_t, \mathbf{x}_{t-1} \in \mathcal{V}^L$. In general, such a kernel would be a $|\mathcal{V}|^L \times |\mathcal{V}|^L$ matrix with $|\mathcal{V}|^{2L}$ entries, which is extremely large even for modest vocabulary and sequence length. Therefore, we use a factorized transition matrix $Q_t \in [0, 1]^{|\mathcal{V}| \times |\mathcal{V}|}$ applied independently at each position of the sequence [63], with $(Q_t)_{ij} = q(x_t^i = j \mid x_{t-1}^i = i)$ for $x_t^i, x_{t-1}^i \in \mathcal{V}$. For the sake of brevity, we will abuse the notation slightly and use $x_t \in \{0, 1\}^{|\mathcal{V}|}$ to denote both the one-hot encoding of a single position in the sequence, as well as the token value $x_t \in \mathcal{V}$ at that position, where the exact meaning will be clear from the context. We will use the superscript to denote the position when necessary. For instance, x_t^i will denote the token at position i in sequence $\mathbf{x}_t$. Using this notation, the transition probability of the Markov chain for one position in the sequence is given by:

$$q(x_t | x_{t-1}) = \text{Cat}(x_t; x_{t-1}^\top Q_t),$$

where $\text{Cat}(\cdot; \theta)$ denotes the categorical distribution with category probabilities θ .

Marginal Distribution: Just as we did for the continuous case in Equation 50.4, we can derive a closed-form expression for the marginal distribution $q(x_t|x_0)$ by marginalizing out all intermediate steps. By the law of total probability and the Markov property:

$$\begin{aligned} q(x_t|x_0) &= \sum_{x_{1:t-1}} q(x_t, x_{t-1}, \dots, x_1|x_0) \\ &= \sum_{x_{1:t-1}} \prod_{s=1}^t q(x_s|x_{s-1}). \end{aligned} \quad (50.30)$$

To evaluate this expression recursively, consider

$$q(x_2 = i|x_0 = j) = \sum_{k=1}^K q(x_2 = i|x_1 = k) q(x_1 = k|x_0 = j) = \sum_{k=1}^K (Q_1)_{jk} (Q_2)_{ki}.$$

This is precisely the matrix multiplication $(Q_1 Q_2)_{ji}$. By induction, we obtain:

$$q(x_t|x_0) = \text{Cat}(x_t; x_0^\top \bar{Q}_t), \quad \text{where} \quad \bar{Q}_t = Q_1 Q_2 \cdots Q_t. \quad (50.31)$$

Posterior Distribution $q(x_{t-1}|x_t, x_0)$: Using the Bayes' rule and the Markov property, as we did for the continuous case in Equation 50.6, we get:

$$\begin{aligned} q(x_{t-1} = i|x_t = j, x_0 = k) &= \frac{q(x_t = j|x_{t-1} = i) q(x_{t-1} = i|x_0 = k)}{q(x_t = j|x_0 = k)} \\ &= \frac{(Q_t)_{ij} (\bar{Q}_{t-1})_{ki}}{(\bar{Q}_t)_{kj}}. \end{aligned}$$

In vectorized form, where $\odot$ denotes element-wise multiplication:

$$q(x_{t-1}|x_t, x_0) = \text{Cat} \left(x_{t-1}; \frac{Q_t^\top x_t \odot \bar{Q}_{t-1}^\top x_0}{x_t^\top \bar{Q}_t^\top x_0} \right). \quad (50.32)$$

Next, we will look at one particular choice of the noising process.

Masking Noise

A particularly effective noising process for discrete data is *masking*, where tokens are gradually replaced with a special [MASK] token. Let m denote the index of the mask token in the vocabulary. The transition matrix is defined as:

$$Q_t = (1 - \beta_t) \mathbf{I} + \beta_t \mathbf{1} \mathbf{e}_m^\top,$$

where $\mathbf{e}_m$ is the one-hot vector for the mask token, $\mathbf{I}$ is the identity matrix, and $\mathbf{1}$ is the vector of all ones, and $\beta_t \in [0, 1]$ is the masking rate at timestep t .

Note that

$$\begin{aligned} Q_s Q_{s+1} &= [(1 - \beta_s) \mathbf{I} + \beta_s \mathbf{1} \mathbf{e}_m^\top] [(1 - \beta_{s+1}) \mathbf{I} + \beta_{s+1} \mathbf{1} \mathbf{e}_m^\top] \\ &= (1 - \beta_s)(1 - \beta_{s+1}) \mathbf{I} + [1 - (1 - \beta_s)(1 - \beta_{s+1})] \mathbf{1} \mathbf{e}_m^\top. \end{aligned}$$

Therefore, through induction:

$$\bar{Q}_t = \bar{\alpha}_t \mathbf{I} + (1 - \bar{\alpha}_t) \mathbf{1} \mathbf{e}_m^\top, \quad (50.33)$$

where, as before, $\bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s)$. As with the case with the continuous domain, we can compute the transition probability $q(x_t|x_0)$ as well as sample from it at a $O(1)$ time complexity if we store the values of $\bar{\alpha}_t$ for $t \in \{1, \dots, T\}$.

Reverse Process

The reverse process must now learn to invert the noising Markov chain. It is parameterised as a categorical distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which, given a noised sequence $\mathbf{x}_t$, predicts the probability distribution over one step of denoising. Parameterising the full joint distribution over $\mathbf{x}_{t-1}$ will require an expensive procedure like Gibbs sampling. Therefore, we approximate the joint distribution using marginals as follows [780]:

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \sum_{\mathbf{x}_0} p(\mathbf{x}_{t-1}, \mathbf{x}_0|\mathbf{x}_t) \\ &= \prod_j \sum_{x_0^j} p(x_{t-1}^j|x_0^j, \mathbf{x}_t) p(x_0^j|\mathbf{x}_t) \\ &\approx \prod_j \sum_{x_0^j} q(x_{t-1}^j|x_0^j, \mathbf{x}_t) p(x_0^j|\mathbf{x}_t) \\ &= \prod_j \sum_{x_0^j} q(x_{t-1}^j|x_0^j, x_t^j) p(x_0^j|\mathbf{x}_t). \end{aligned}$$

Using this, we can parameterise the reverse process as

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \prod_{i=1}^L p_\theta(x_{t-1}^i|\mathbf{x}_t), \quad \text{where,} \quad p_\theta(x_{t-1}^i|\mathbf{x}_t) = \sum_{x_0^i} q(x_{t-1}^i|x_0^i, x_t^i) p_\theta(x_0^i|\mathbf{x}_t), \quad (50.34)$$

where we can insert the posterior $q(x_{t-1}^i|x_0^i, x_t^i)$ in closed form from Equation 50.32.

Objective

Since the ELBO in Equation 50.11 was obtained with no additional assumptions on the q and p , it remains fully applicable even for the discrete case. In the case of masked noising, injecting the simple posterior from Equation 50.33 into Equation 50.34, we get:

$$p_\theta(x_{t-1}^i|\mathbf{x}_t) = \begin{cases} 1, & \text{if } x_t^i = x_0^i \text{ and } x_{t-1}^i = x_0^i \\ 0, & \text{if } x_t^i = x_0^i \text{ and } x_{t-1}^i \neq x_0^i \\ \frac{\beta_t \prod_{s=1}^{t-1} (1-\beta_s)}{1 - \prod_{s=1}^{t-1} (1-\beta_s)} p(x_0^i = x_{t-1}^i|\mathbf{x}_t), & \text{if } x_t^i = m \text{ and } x_{t-1}^i \neq m \\ \frac{1 - \prod_{s=1}^{t-1} (1-\beta_s)}{1 - \prod_{s=1}^{t-1} (1-\beta_s)}, & \text{if } x_t^i = m \text{ and } x_{t-1}^i = m. \end{cases} \quad (50.35)$$

Therefore, we just need to parameterise $p_\theta(x_0^i = x_{t-1}^i|\mathbf{x}_t)$ for the case when $x_t^i = m$. This observation simplifies L_{VLB} from Equation 50.11 into

$$L_{\text{VLB}}(\theta) = -\mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_t} \frac{\beta_t \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \sum_{i=1}^L \mathbb{1}(x_t^i = [\text{MASK}]) \log p_\theta(x_0^i|\mathbf{x}_t),$$

where the expectation is over $t \sim \text{Uniform}[0, T]$, $\mathbf{x}_0 \sim q_{\text{data}}(\mathbf{x}_0)$ and $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$. This objective is just a weighted sum of cross-entropy losses again and is extremely simple to compute.

Extension to Continuous-Time Formulation

As was the case with diffusion in continuous space, the discrete-time formulation in Section 50.7.2 can be extended to continuous time. For the continuous-time process, $\mathbf{Q}_t$ will denote the *rate matrix*, which can be thought of as the infinitesimal version of the transition matrix, but with real valued entries. The off-diagonal entries of $\mathbf{Q}_t$ are non-negative, and the diagonals are the negative sum of the off-diagonal entries: $(\mathbf{Q}_t)_{ij} \geq 0$ for $i \neq j$ and $(\mathbf{Q}_t)_{ii} = -\sum_{j \neq i} (\mathbf{Q}_t)_{ij}$. Using $\mathbf{q}_t$ to denote the vector of marginals, the evolution of the marginals is governed by the Kolmogorov forward equation (KFE):

$$\frac{d}{dt} \mathbf{q}_t^\top = \mathbf{q}_t \mathbf{Q}_t, \quad \mathbf{q}_0 = \mathbf{q}_{\text{data}}. \quad (50.36)$$

The most intuitive way to understand the evolution of the continuous-time Markov chain $\mathbf{x}_t$ is by considering the discretised form of the ODE for small Δt :

$$q(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t) = \mathbb{1}(\mathbf{x}_{t+\Delta t} = \mathbf{x}_t) + \Delta t \mathbf{Q}_t(\mathbf{x}_t, \mathbf{x}_{t+\Delta t}) + o(\Delta t). \quad (50.37)$$

Next, we will discuss the factorized masking process governed by the per-position rates $Q_t \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$.

Masking Noise: The rate matrix of the continuous-time analog of token masking noising (Section 50.7.2) can be written using a single function of time $\beta(t) \geq 0$ (which we will denote by β_t):

$$Q_t = -\beta_t \mathbf{I} + \beta_t \mathbf{1} \mathbf{e}_m^\top. \quad (50.38)$$

In order to compute the transition probability $q(x_t|x_0)$ we observe that the rate matrices for different time values commute. Therefore, we get a closed-form solution for the Kolmogorov forward equation, which gives us the transition probability in closed form:

$$q(x_t|x_0) = \text{Cat}(x_t; \bar{\alpha}_t x_0 + (1 - \bar{\alpha}_t) \mathbf{e}_m^\top), \quad \text{where} \quad \bar{\alpha}_t = \exp\left(-\int_0^t \beta_s ds\right), \quad (50.39)$$

and, as before, x_0 is the one-hot vector for the starting state, $\mathbf{e}_m$ the one-hot vector for the mask token. Notice that the transition probability mirrors the discrete-time formula closely, with the product of the discrete time schedule $\prod_k (1 - \beta_k)$ replaced by the exponential of the integral of the function β_t .

Reverse Process: The reverse of the Markov noising process is also a Markov process. Moreover, the rate matrix of the reverse process, $\tilde{Q}_t$ can be written in terms of the rate matrix of the forward process:

$$(\tilde{Q}_t)_{ji} = (Q_t)_{ij} \frac{q(x_t = i)}{q(x_t = j)} \quad \text{for } i \neq j. \quad (50.40)$$

The advantage of using continuous-time is that we can parameterise the per-position marginals of the reverse process without approximation because the probability of two events (e.g. two token changes) occurring at the same time is zero in continuous time.

There are several choices of parameterisations for the reverse process. We can parameterise the per-position rates $\tilde{Q}_{\theta,t}^i(\mathbf{x}_t, y^i)$ directly and set the overall rate to

$$\tilde{\mathbf{Q}}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \sum_{i=1}^L \mathbb{1}(x_t^{-i} = y^{-i}) \tilde{Q}_{\theta,t}^i(\mathbf{x}_t, y^i).$$

Lou et al. [1082] uses the known relationship in Equation 50.40 and parameterise the unknown marginal density ratio $\frac{[q(\mathbf{y})]_i}{q(\mathbf{x}_t)_i}$ using a transformer, where the transformer produces a positive real value for each position i and token y^i in the vocabulary. Notice that the role of this density ratio is similar to that of the *score function* in the continuous diffusion, which acts as the drift that moves the particles of the reverse process towards the data distribution. Another, more standard approach [229] is to parameterise the data probability $p_\theta^i(x_0^i|\mathbf{x}_t)$ directly using the softmax output of a transformer, and set the reverse rate to

$$\tilde{\mathbf{Q}}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \sum_{i=1}^L Q_t^i(y^i, x_t^i) \mathbb{1}(x_t^{-i} = y^{-i}) \sum_{x_0^i} \frac{q(y^i|x_0^i)}{q(x_t^i|x_0^i)} p_\theta^i(x_0^i|\mathbf{x}_t) \quad (50.41)$$

In the case of masking noise, this parameterisation leads to a simple training objective, which is a weighted sum of cross-entropy losses [1446, 1514]:

$$L_{\text{VLB}}(\theta) = -\mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_t} \frac{\alpha_t}{1 - \bar{\alpha}_t} \sum_{j=1}^L \mathbb{1}(x_t^j = [\text{MASK}]) \log p_\theta(x_0^j|\mathbf{x}_t). \quad (50.42)$$

Note that the continuous-time weight $\alpha_t/(1 - \bar{\alpha}_t)$ differs from the discrete-time form in Equation 50.11 above.

Sampling: Once we have the learned rate matrix for the reverse process ($\tilde{\mathbf{Q}}_{\theta,t}$), the simplest sampling method is to discretise the continuous-time process and simulate the transitions according to Equation 50.37. For time-inhomogeneous Markov chains, as in our case, this naive approach requires extremely small time steps to ensure accuracy. One can use more sophisticated methods like Gillespie’s algorithm [631], which do not require explicit discretisation of time to simulate the process. However, these exact algorithms can be quite slow as they require stepping through each transition individually. In practice, it has been observed that for many applications, one can use fast approximate simulation based on Tau-leaping [630], where the time is discretised, but multiple transitions are performed in a single step. That is, multiple tokens in a sequence change their values simultaneously after a single evaluation of the neural network. The success of simultaneous sampling from masked diffusion models has led to a resurgence of interest in these models for applications such as code generation, where they outperform autoregressive models in terms of speed.

50.8 Chapter Summary

Denoising diffusion models have established themselves as a new pillar of generative modelling, delivering state-of-the-art sample quality with a stable and principled training objective.

- Their core mechanism consists of a fixed forward process q that adds noise and a learned reverse process p_θ that removes it.
- Their training is grounded in the variational inference framework (ELBO), sharing a theoretical lineage with VAEs.
- The central insight of DDPMs is the simplification of the ELBO to a weighted L_2 loss, which trains a network ϵ_θ to predict the noise added at a given timestep t .
- This discrete-time formulation is elegantly unified with score-based modelling through the continuous-time SDE framework, where ϵ_θ is shown to be a reparameterised score function. This view opens up flexible sampling solvers, including SDE, ODE, and predictor-corrector methods.
- Practical utility is enhanced by algorithms for sampling acceleration (DDIM) and precise conditional generation (classifier-free guidance, CFG).
- The framework is flexible and generalisable from continuous domains (images, audio) to discrete domains (text, graphs) by replacing the Gaussian perturbation kernel with a discrete transition matrix.

50.9 Bibliographical Notes and Further Reading

The concept of diffusion for generative modelling was first proposed by Sohl-Dickstein et al. [1532], framing it in the context of non-equilibrium thermodynamics. The modern resurgence was sparked by the work of Ho et al. [755], who introduced the specific parameterisations (DDPM) and simplifications that made these models practical and state-of-the-art. Independently, Song and Ermon [1540] developed a parallel track of score-based generative models, which was later unified with DDPMs in the SDE framework by Song et al. [1541].

Key practical advancements include the development of Denoising Diffusion Implicit Models (DDIM) for fast sampling by Song et al. [1538], and the introduction of classifier-free guidance (CFG) by Ho and Salimans [753], which is now the standard for conditional image generation. Latent diffusion models were introduced by Vahdat et al. [1650] and later improved upon by Rombach et al. [1417]. Subsequent work refined the training and conditioning recipe for large-scale text-to-image generation [1333], and replaced the latent U-Net denoiser with a transformer operating on latent patches [1309].

A complementary line of work that aims to reduce the number of sampling steps is one-step and few-step generation. The literature in this area has developed along several related directions. A useful starting point is the work of Song et al. [1542] on consistency models, which introduces functions that remain invariant along trajectories of the probability-flow ODE and can therefore map noise to data in a single evaluation, while also supporting multistep sampling when additional computation is available. For approaches based more explicitly on distillation, see distribution matching distillation (DMD) [1817], which trains a one-step generator using differences between the scores of the target and generated distributions. Score identity distillation [1871] and its adversarial extension, SiDA [1872], provide a related perspective based on Fisher-divergence objectives and the use of real-data adversarial supervision. MeanFlow offers a complementary route through flow-based modeling, wherein rather than

numerically integrating an instantaneous velocity field over many steps, it learns the average velocity over a finite time interval, making one-step transport possible [605]. DiffRatio is also relevant in this context, especially for readers interested in density-ratio formulations. It estimates ratios between noised data and model distributions, whose log-gradients recover the score differences needed to train a one-step generator without direct teacher-score supervision [276].

Discrete diffusion models also have a long history. The original formulation of Sohl-Dickstein et al. [1532] included a binomial diffusion process for binary data. A more general DDPM-style treatment of structured categorical state spaces was developed by Austin et al. [63], whose discrete denoising diffusion probabilistic models (D3PMs) support uniform, absorbing-state, and structure-aware transition kernels. Campbell et al. [229] subsequently formulated the forward and reverse processes as continuous-time Markov chains, providing a general continuous-time framework for discrete denoising models. Several later works improved the objectives used to train discrete diffusion models for language. Lou et al. [1082] introduced score-entropy discrete diffusion, which parameterises the reverse process through ratios of probabilities under the noised data distribution. For absorbing-state or masked diffusion, Shi et al. [1514] showed that the continuous-time variational objective reduces to a weighted integral of cross-entropy losses. Independently, Sahoo et al. [1446] derived a Rao-Blackwellised objective expressible as a mixture of masked-language-modeling losses and demonstrated that masked diffusion language models can approach autoregressive perplexity when trained with an appropriate architecture and optimization recipe. For the uniform-state setting, the Diffusion Duality (Duo) relates discrete diffusion to an underlying Gaussian process and uses this connection to adapt techniques from continuous diffusion, including consistency distillation for few-step generation [1447]. Variable-length generation requires models whose state space includes sequences of different lengths. Recent literature has developed insertion-based generative models, wherein insertion of tokens is viewed as the *denoising* operation [1300]. This idea has also been extended to use continuous-time using continuous-time Markov chains in which the forward noising process deletes elements and generative process inserts them [428, 1301]. EditFlows generalize the idea to allow for insertions, deletions, and substitutions within a continuous-time flow-matching framework [716]. Flexible length masked diffusion (FlexMDM) offers a related construction in which sequence length changes through the insertion of mask tokens that are subsequently denoised [887].

Chapter 51

Prior-data Fitted Networks

Chapter 3 introduced Simulation-Based Inference (SBI) as a powerful framework for performing Bayesian inference $p(\theta | X)$ when we cannot measure the likelihood $p(X | \theta)$, but only sample from it. However, in many scenarios, one may not be interested in estimating the latent parameters θ , but seek to make supervised probabilistic predictions. In a supervised prediction setting we have a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ of pairs of input features and labels, and our task is to predict the label for features x_{test} based on $\mathcal{D}$. We are thus primarily interested in the posterior predictive distribution (PPD) $p(y_{\text{test}} | x_{\text{test}}, \mathcal{D})$. The PPD is typically approximated using the posterior over the latent variables $p(\theta | \mathcal{D})$ in Bayesian prediction. In this chapter, we instead directly amortise the computation of the PPD without explicitly computing a posterior over latent variables. Müller et al. [1208] proposed this approach, including a parameterisation using neural networks, as *Prior-data Fitted Networks* (PFNs)¹.

A PFN is any neural network that amortises the posterior over the label y_{test} , unlike SBI which amortises the posterior inference of the latent variable θ .

51.1 From Inference to Prediction

To understand the connection between inference of latent variables θ , and the prediction of y_{test} , note that the PPD can be defined using the posterior of θ as

$$p(y_{\text{test}} | x_{\text{test}}, \mathcal{D}) = \int p(y_{\text{test}} | x_{\text{test}}, \theta) p(\theta | \mathcal{D}) d\theta, \quad (51.1)$$

where θ are the latent variables, and $p(\theta | \mathcal{D})$ is the posterior over θ given the training data.

Using SBI (or any other posterior inference method described in this book), we can formulate a two-step process to approximate the PPD:

1. Inference: Use the inference method to obtain an approximate posterior $q(\theta | \mathcal{D}) \approx p(\theta | \mathcal{D})$.
2. Prediction: Approximate the PPD integral by drawing samples $\theta_j \sim q(\theta | \mathcal{D})$ and averaging their predictions (i.e., a Monte Carlo estimate):

$$p(y | x, \mathcal{D}) \approx \frac{1}{M} \sum_{j=1}^M p(y | x, \theta_j) \quad (51.2)$$

The inference step can be inaccurate, as q has to typically model a complex multivariate distribution, and the prediction step requires sampling many times from the approximate posterior q_ψ .

In this chapter, we make this a single step instead: One can bypass the explicit estimation of the latent posterior $p(\theta | \mathcal{D})$ and instead learn an amortised model that directly approximates the posterior predictive distribution $p(y | x, \mathcal{D})$. PFNs do this by training a neural network to make the right predictions in one forward pass. This is still amortising, just like SBI, but the amortisation now targets the label and not the latent parameters. Instead of training a model to output the posterior over the latent, the model is trained to output the predictive posterior for a supervised learning problem.

¹The idea was previously explored in some preliminary experiments by Garnelo et al. [582].

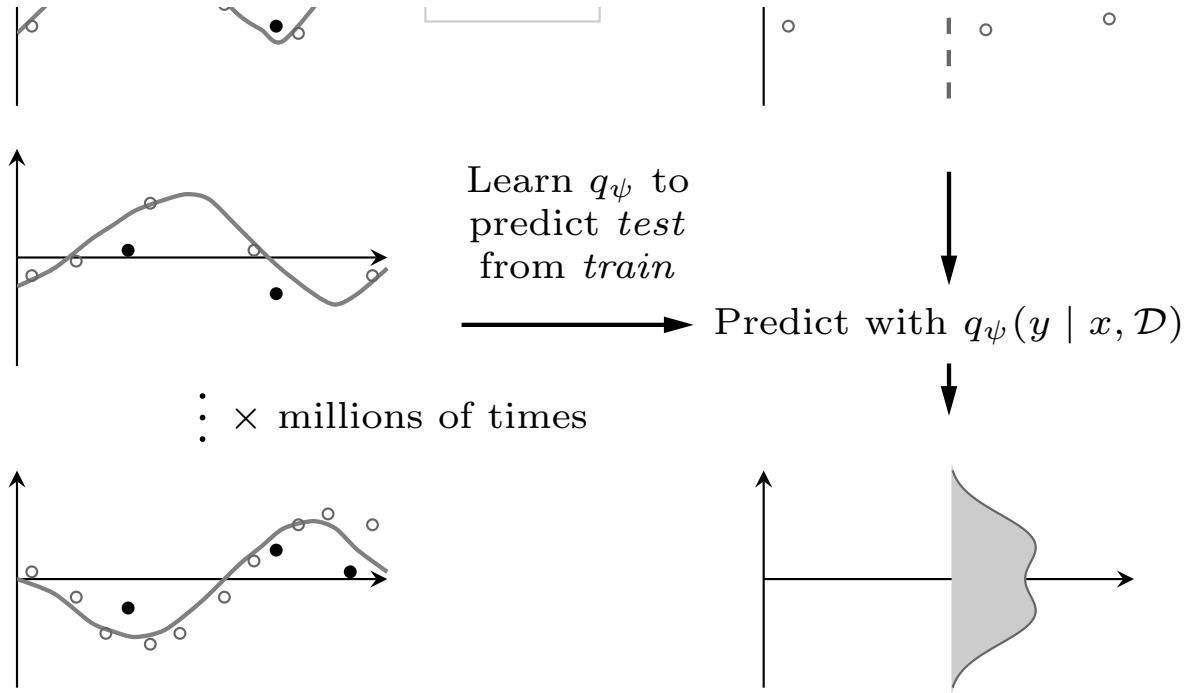


Figure 51.1: An outline of prior-fitting and prediction on a 1D Gaussian Process prior. We first fit the model q_ψ to predict held-out labels on sampled datasets. Then, we use this model to make predictions on new real-world data by feeding it the context. On the bottom right we see the PDF of the model's output, its posterior approximation.

51.2 Prior Fitting

PFNs are trained on entire datasets sampled from the prior. The core idea is simple: one generates a large number of synthetic datasets from the prior and the PFN is fit to predict held-out labels given all the remaining data for all datasets. We illustrate this process in Figure 51.1. Building dataset priors is possible for a very large class of priors, as we describe in the next section.

More formally, a PFN q_ψ with parameters ψ is trained to minimise the cross-entropy loss

$$\mathcal{L}(\psi) = \mathbb{E}_{\mathcal{D} \cup \{(x_{\text{test}}, y_{\text{test}})\} \sim p(\mathcal{D})} [-\log q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D})], \quad (51.3)$$

where $p(\mathcal{D})$ is a prior over datasets. This objective alone is enough to ensure that the PFN approximates the PPD.

The objective $\mathcal{L}(\psi)$ is similar to the NPE loss in Equation (3.3), but targets the predictive distribution directly rather than the posterior over latents (see Chapter 3 for NPE, NLE, and NRE). We can show that the learned model converges to the true PPD under appropriate conditions (sufficient model capacity, well-specified prior, convergence to the global minimum) [541, 1208]. To do this, we can rewrite the loss as

$$\begin{aligned} \mathcal{L}(\psi) &= \mathbb{E}_{\mathcal{D} \cup \{(x_{\text{test}}, y_{\text{test}})\} \sim p(\mathcal{D})} [-\log q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D})] \\ &= \mathbb{E}_{\mathcal{D}, x_{\text{test}} \sim p(\mathcal{D}, x_{\text{test}})} \mathbb{E}_{y_{\text{test}} \sim p(y_{\text{test}} | x_{\text{test}}, \mathcal{D})} [-\log q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D})] \\ &= \mathbb{E}_{\mathcal{D}, x_{\text{test}} \sim p(\mathcal{D}, x_{\text{test}})} \int -p(y_{\text{test}} | x_{\text{test}}, \mathcal{D}) \log q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D}) dy_{\text{test}} \\ &= \mathbb{E}_{\mathcal{D}, x_{\text{test}} \sim p(\mathcal{D}, x_{\text{test}})} H(p(\cdot | x_{\text{test}}, \mathcal{D}), q_\psi(\cdot | x_{\text{test}}, \mathcal{D})) \\ &= \mathbb{E}_{\mathcal{D}, x_{\text{test}} \sim p(\mathcal{D}, x_{\text{test}})} \text{KL}(p(\cdot | x_{\text{test}}, \mathcal{D}) || q_\psi(\cdot | x_{\text{test}}, \mathcal{D})) + C, \end{aligned} \quad (51.4)$$

where C is a constant that is independent of q_ψ and H denotes cross-entropy. Minimising this loss thus means minimising the KL-divergence to the posterior on tasks sampled from the prior. A distribution that has a KL-divergence of 0.0 with another distribution is the same as the other distribution almost everywhere. Therefore, minimising this loss yields $q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D}) \approx p(y_{\text{test}} | x_{\text{test}}, \mathcal{D})$ for almost all $(x_{\text{test}}, \mathcal{D})$ in the support of the prior.

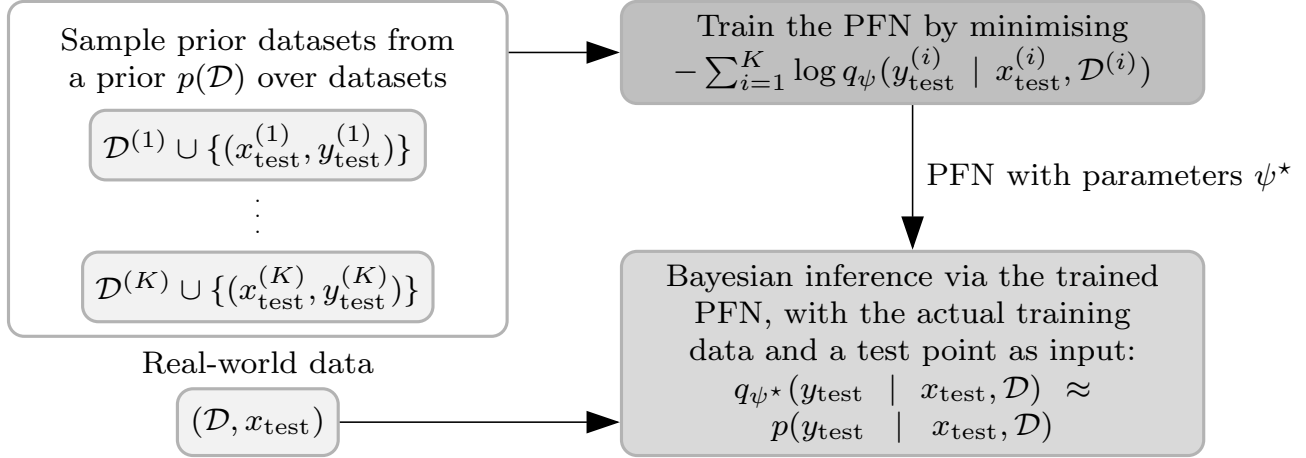


Figure 51.2: This figure outlines the components of the PFN training process more abstractly. We approximate Equation 51.3 by sampling datasets from the prior in small batches and the process yields a model that approximates the PPD.

The objective $\mathcal{L}(\psi)$ is optimised using standard deep learning techniques: sampling batches of datasets and performing stochastic gradient descent on them using Adam [894]. We outline the training process and the resulting model’s behaviour in Figure 51.2.

51.3 Dataset Priors

To train PFNs, we need to define priors over datasets $p(\mathcal{D})$. In many scenarios, an applicable Bayesian model $f_{\theta}(x)$ mapping x to an output, or more generally, a likelihood $p(y|x, \theta)$ with parameters θ sampled from prior $p(\theta)$ already exists, but its inference has generally been carried out with traditional Bayesian methods. In these cases, a prior over datasets can be defined as $p(\mathcal{D}) = \int p(\theta) \prod_{0 < i \leq N} p(x_i)p(y_i|x_i, \theta)d\theta$, where $p(x)$ is usually chosen to be easy to sample and have wide coverage, e.g., $x \sim \mathcal{N}(0, 1)$. Note that defining the prior using the conditional distribution $p(y|x, \theta)$ is a common technique, but not the only one, e.g., TabPFNs define $p(\mathcal{D}|\theta)$ directly. The process to sample one dataset based on a latent prior $p(\theta)$ can be outlined as:

1. First, sample a latent parameter vector $\theta \sim p(\theta)$.
2. Second, sample a full dataset $\mathcal{D} = \{(x_i, y_i) \sim p(x, y \mid \theta)\}_{i=1}^N$ from the likelihood defined by θ . If θ only defines the likelihood $p(y|x, \theta)$, one typically samples x from some uninformative distribution.

This prior $p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta) d\theta$ defines the assumptions about the kinds of datasets we expect to see. Critically, just as with SBI, the approach only requires sampling from the prior and the likelihood. This allows modellers to define rich and complex priors, such as Bayesian neural networks (BNNs) with uncertainty over their architecture, which we outline later in this section.

Most of the PFN priors used thus far are inspired by priors in traditional Bayesian inference setups, such as Bayesian neural networks or Gaussian processes, thus rely on a latent sampling setup as explained above. PFNs tend to have wider priors compared to traditional methods, though, as they both: i) work with latents that are really hard to model directly, like the architecturally uncertain BNN above, and ii) need to be expensively retrained for every change of prior, thus making it important to choose priors that generalise across tasks. This is, e.g., the case for the TabPFNs [769, 770] or for PFNs4BO [1209], which target whole data modalities (tabular classification, regression and Bayesian optimisation). Nevertheless, there are also domain-specific PFNs, similar to NPEs, e.g., to predict RNA folding times [1469].

As examples of typical priors used with PFNs, we detail two very simple, educational priors and their more practical extensions. The priors we look into are (i) Bayesian linear regression / neural network priors and (ii) Gaussian process (GP) priors.

51.3.1 A Bayesian Linear Regression Prior

Bayesian linear regression provides one of the simplest yet most instructive priors for understanding PFNs.

For educational purposes, we define a Bayesian linear regression prior, whose mean corresponds to ridge regression predictions with regularisation parameter $\alpha \in \mathbb{R}$. The prior uses another parameter σ^2 to control the noise level relative to α . A dataset with N samples and F features can be sampled from this prior using the following steps:

1. Sample weights $w \in \mathbb{R}^F$ as $w_i \sim \mathcal{N}(0, \sigma^2/\alpha)$.
2. Sample a bias $b \sim \mathcal{N}(0, \sigma^2/\alpha^{-1})$.
3. Sample inputs $X \in \mathbb{R}^{N \times F}$ independently from $\mathcal{N}(0, 1)$ or $\mathcal{U}(0, 1)$.
4. Compute the noiseless predictions $\mu = Xw + b$.
5. Sample noisy targets $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$ for $i = 1, \dots, N$.
6. Return the dataset $\{(X_i, y_i)\}_{i=1}^N$.

To clarify the connection to the expected PPD, we write out the underlying likelihood and parameter prior. The parameter prior is given by

$$p(w_0, \dots, w_{F-1}, b) = \left(\prod_{0 \leq i < F} f_{\mathcal{N}(0, \sigma^2/\alpha^{-1})}(w_i) \right) \cdot f_{\mathcal{N}(0, \sigma^2/\alpha^{-1})}(b), \quad (51.5)$$

where $f_{\mathcal{N}(\mu, \sigma^2)}$ denotes the density of the normal distribution with mean μ and variance σ^2 . The likelihood is defined as

$$p(y|x, w, b) = f_{\mathcal{N}(x^T w + b, \sigma^2)}(y). \quad (51.6)$$

Going from these definitions to the prior sampling procedure boils down to the recipe given in the beginning of the section: sample from the prior $p(w_0, \dots, w_{F-1}, b)$ first, sample inputs X (in an uninformative fashion), and finally sample the targets y in our dataset from $p(y|X_i, w, b)$ for $0 \leq i < N$.

Further, it can be shown that the mean of the Bayesian linear regression predictions actually coincides with the ridge regression predictions, as ridge regression is a MAP estimate on the same prior, and the MAP coincides with the mean posterior as the posterior is a normal distribution; further, as the model is linear, one can factor out the posterior when computing an expectation over latents to get the PPD. Thus, the predictions of a PFN trained on this prior will in the limit have the same mean as the ridge regression prediction. We leave the exact proof as an exercise.

51.3.2 A Bayesian Neural Network Prior with Architectural Uncertainty

In this section, we define a Bayesian neural network (BNN) prior as an extension of the ridge regression prior presented above, which can be seen as a one-layer BNN prior without hidden layers. Notably, PFNs trained on the following prior extend traditional BNNs, which primarily quantify the uncertainty in the network weights, by also quantifying the uncertainty over architectural choices and weight-sparsity patterns. This is difficult to achieve with traditional methods such as VI (Chapter 12) or MCMC (Chapter 1) as it would involve instantiating many neural network architectures at prediction time. For PFNs, on the other hand, it is just a matter of designing a dataset prior. This is a prior that is practically relevant, as slight modifications of it were successfully used for tabular prediction tasks [769] and Bayesian optimisation [1209].

The prior can be described by the following dataset sampling procedure, where the example values are taken from Müller et al. [1209] wherever possible.

1. First, sample the architecture of the neural network modelling the dataset.
 - (a) Sample the number of layers $L \sim \mathcal{U}(\{L_{\min}, \dots, L_{\max}\})$ (e.g., $L_{\min} = 8$, $L_{\max} = 15$).
 - (b) Sample the number of units used in all hidden layers $H \sim \mathcal{U}(\{H_{\min}, \dots, H_{\max}\})$ (e.g., $H_{\min} = 36$, $H_{\max} = 150$).
2. Sample the weights.
 - (a) First, the standard deviation of all weights across the network is sampled as $\sigma \sim \mathcal{U}(\sigma_{\min}, \sigma_{\max})$ (e.g., $\sigma_{\min} = 0.1$, $\sigma_{\max} = 0.2$).

- (b) Then, all weight matrices θ_l and bias vectors θ'_l in layers $0 \leq l < L$ are sampled independently from $\mathcal{N}(0, \sigma^2)$.
- (c) Finally, the layers are sparsified by setting each weight to 0 with sparsity probability p_{sparse} (e.g., $p_{\text{sparse}} = 14.5\%$).
- 3. Sample inputs $X \in \mathbb{R}^{N \times F}$ independently and uniformly at random in $\mathcal{U}(0, 1)$, where N is the total number of samples in the training set and F is the number of features.
- 4. Pass the inputs as a batch through the neural network, defined in steps 1 and 2, to retrieve the output vector $y = \theta_{L-1} \text{ReLU}(\theta_{L-2} \text{ReLU}(\dots \text{ReLU}(\theta_0 X + \theta'_0) \dots) + \theta'_{L-2}) + \theta'_{L-1}$, where $y \in \mathbb{R}^N$.
- 5. Consolidate the dataset as $\{(X_i, y_i)\}_{0 \leq i < N}$.

We can now train the PFN on datasets generated like this by holding out a part of any given dataset as a test set and using the remaining part as the conditioning set. In doing so, the PFN learns to approximate Bayesian inference over the BNN, including its architecture and weight-sparsity pattern, in addition to the values of the weights. Notably, the theoretical justification behind this posterior approximation only holds for $X \in [0, 1]$, as the model is not trained on data outside of this range.

51.3.3 SCM Extension

The BNN from the previous section (Section 51.3.2) can now easily be further extended to a structural causal model (SCM) prior. SCM priors are generalisations of BNNs and widely used in tabular foundation models [769, 770, 1861]. SCMs are directed acyclic graphs of arbitrary shapes, where the value of a node is defined by some deterministic function of its ancestors' values and some noise.

The SCM definition is thus very broad, so even the BNN prior we presented previously could be seen as a narrow SCM prior, where each node of the graph is a neuron in the neural network, it always has the features as roots and the output as the single leaf of this graph. The functions are simply the linear layers, activations and noises.

To build a more proper SCM prior that allows for randomness and more complex dependencies, we can make the following two changes to the prior:

i) First, we add noise to all activations in the neural network. We can write the network now as $h_i = \text{ReLU}(\theta_i h_{i-1} + \theta'_i) + \epsilon_i$, where $h_{-1} = X$ and $y = \theta_{L-1} h_{L-2} + \theta'_{L-1}$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$. ii) Second, and most importantly, we get rid of the streamlined structure. Now when building the dataset, we don't use X as features and y as our target but sample both uniformly out of all activations $h_{i,j}$.

These two simple interventions turn the prior presented above into an SCM prior with complex interdependencies, e.g., one feature might be depending on another feature or the target. One potential problem that we did not encounter so far, though, is that we can't sample from $p(y|x, \theta)$ as y might be represented by an activation that is earlier in the graph (neural network) than some of the features in x . This is not a problem for PFN training, though, as all we need to do is sample from $p(x, y|\theta)$ to build a dataset with shared latents $\theta \sim p(\theta)$. We simply sample our data as before, but read off different values in the computational graph to build the dataset.

51.3.4 A Simple Gaussian Process Prior

We first consider a simple GP prior with fixed hyperparameters (for more background on this prior see Chapter 23), before introducing a more complex GP prior that is applicable to real-world data below. The following simple GP prior with fixed hyperparameters is supposed to provide an intuition for how kernels can be used as dataset priors.

Datasets with F features and N samples can be sampled from a simple RBF GP prior with length-scale ℓ and a noise variance of σ^2 as:

1. Sample inputs $X \in \mathbb{R}^{N \times F}$ uniformly from $\mathcal{U}(0, 1)^F$.
2. Build the covariance matrix $\Sigma \in \mathbb{R}^{N \times N}$ using an RBF kernel:

$$\Sigma_{i,j} = k(X_i, X_j) = \exp\left(-\frac{r_{i,j}^2}{2\ell^2}\right), \quad (51.7)$$

where $r_{i,j} = \|X_i - X_j\|$ is the Euclidean distance between inputs.

3. Add observation noise to the diagonal: $\tilde{\Sigma} = \Sigma + \sigma^2 \mathbf{I}_N$.
4. Sample targets from the multivariate normal: $y \sim \mathcal{N}(0, \tilde{\Sigma})$.
5. Return the dataset $\{(X_i, y_i)\}_{i=1}^N$.

This simple prior will converge to behave like the GP with the same hyperparameters [1208, 1209]. It is very limited, though, as the hyperparameters that, e.g., control how steep the posterior can be, are fixed.

51.3.5 A Fully Bayesian Gaussian Process Prior

Previously, we outlined a simple GP prior with fixed kernel hyperparameters for educational purposes. We now extend this example to yield the more complex HEBO⁺ prior presented by Müller et al. [1209]. Training on this prior lets PFNs approximate a Gaussian process with uncertainty over the hyperparameters and feature importances, which yields state-of-the-art Bayesian optimisation results [1209].

The sampling process for this prior is still simple and similar to the one above, but introduces some extra sampling steps:

1. First, a subset of the features is sampled to be used and not ignored by flipping a coin for each feature $i_f \sim \text{Ber}(p_{\text{use}})$ (e.g., $p_{\text{use}} = 0.85$), where p_{use} is the probability that a feature is used. We define $F' = \sum_f i_f$.
2. Next, as many length-scales $\ell \sim p(\ell)$ as relevant features ($i_f = 1$), a noise variance $\sigma^2 \sim p(\sigma^2)$, and an output scale $s \sim p(s)$ are sampled from pre-defined prior distributions.
3. Further, we sample the inputs $X \in \mathbb{R}^{N \times F}$ uniformly from $\mathcal{U}(0, 1)$.
4. Now, we build a covariance matrix using the Matérn-3/2 kernel $\Sigma_{i,j} = k(X'_i, X'_j) = \sigma^2 \left(1 + \frac{\sqrt{3}r}{\ell}\right) \exp\left(-\frac{\sqrt{3}r}{\ell}\right)$, where $r = \|X'_i - X'_j\|$ and $X' \in \mathbb{R}^{N \times F'}$ are the relevant features of X .
5. The targets can now be sampled as $y \sim \mathcal{N}(0, \Sigma + \sigma^2 \mathbf{I}_N)$, where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

This process not only trains a PFN to approximate a classical Gaussian process with uncertainty over its hyperparameters, but additionally trains the model to do Bayesian inference over which subsets of features are inactive and can be ignored, a capability previously reserved to expensive Bayesian ensemble models like the sparse axis-aligned subspaces (SAAS) model of Eriksson and Jankowiak [497].

51.4 PFN Architectures

A key challenge in implementing PFNs is designing neural network architectures that work for the set-based PFN tasks. The architecture should be able to accept variable-length training sets and in most cases also variable numbers of features. Further, it is natural to embed invariances into the architecture present in the prediction task:

- (i) **Train invariance:** The predictions should be invariant to the ordering of training samples in $\mathcal{D}$.
- (ii) **Test independence:** When predicting multiple test samples at once, their predictions should be modelled independently of one another.
- (iii) **Variable train size:** The architecture must handle datasets of varying sizes.
- (iv) **Feature invariance:** In most tabular scenarios, the order of the features present in our inputs x should not matter and our models should thus be invariant to feature permutations. There are others, though, where this property is not desirable, e.g., image data.

All requirements, besides **Feature invariance**, are shared with *conditional neural processes* [582]. For their meta-learning context, see Chapter 27.

All PFN architectures to date are based on the Transformer [1669] even though that is not a requirement. In the following, we will discuss two popular types of architectures: the 1-D attention architecture originally proposed together with PFNs and the 2-D attention architecture proposed for TabPFNv2 [770].

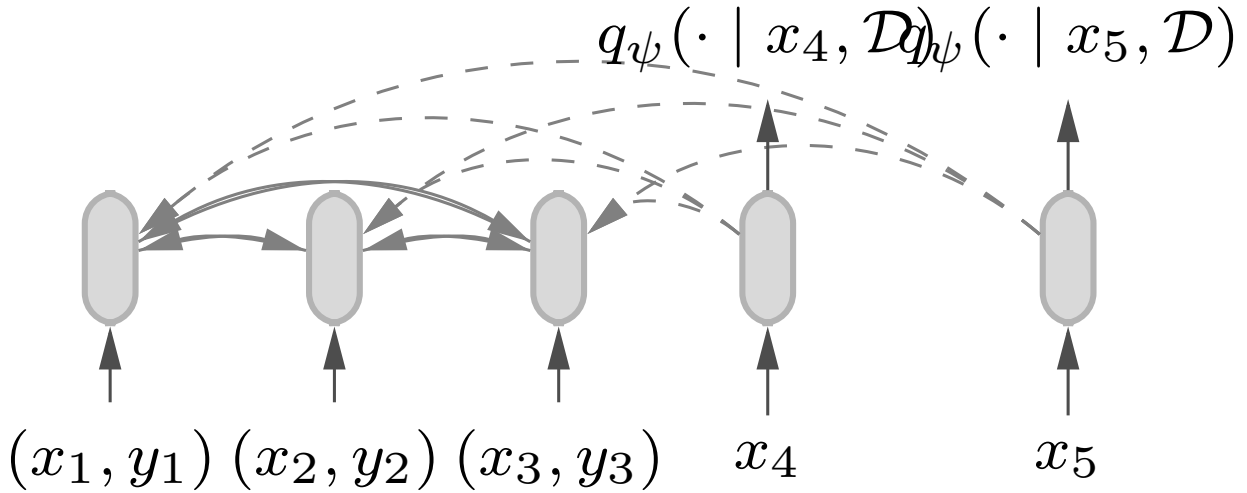


Figure 51.3: Original PFN and TabPFNv1 architecture. Dashed arrows indicate test attention, while solid arrows indicate train attention.

51.4.1 Original PFN Architecture

The original PFN architecture achieved a breakthrough in the accurate modelling of a general set of posteriors, and was most prominently used to power the TabPFNv1 model for tabular data [769]. It leverages a transformer architecture [1669] that was minimally modified to satisfy the invariance requirements (i-iii), akin to the architectural requirements of the conditional neural process, see Figure 51.3 for an illustration. The key modifications to the Transformer encoder are in the attention mechanism: training samples $(x_i, y_i) \in \mathcal{D}$ can attend to each other bidirectionally, but test inputs x_{test} can only attend to the training samples, and not to other test inputs. Further, any positional embeddings are removed. The output of the last layer provides the parameters of the predictive distribution $q_\psi(y_{\text{test}} | x_{\text{test}}, \mathcal{D})$ at each test position x_{test} . Nguyen and Grover [1245]’s Transformer Neural Processes later extended the architecture to additionally approximate the joint PPD of several test points.

We now describe the architecture more formally and with multiple test inputs $\{x_{\text{test}}^{(j)}\}_{j=1}^{n_{\text{te}}}$ that are handled independently of each other by the model. The transformer maintains one state per training sample and per test input. Throughout, we use N for the number of training samples and n_{te} for the number of test inputs $\{x_{\text{test}}^{(j)}\}_{j=1}^{n_{\text{te}}}$, d for the embedding dimension, L for the number of transformer layers, H for the number of attention heads (with per-head width d_h), $e_x(\cdot)$, $e_y(\cdot)$ for affine encoders, $W_{Q,h}^{(\ell)}$, $W_{K,h}^{(\ell)}$, $W_{V,h}^{(\ell)}$, $W_O^{(\ell)}$ for layer parameters and $\text{FFN}^{(\ell)}$ for MLPs, LN for layer normalisation, and for the regression head we use parameters W_h and $\mathbf{b}_h$.

The model first encodes the raw dataset using two simple linear (affine) encoders $e_x(\cdot)$ and $e_y(\cdot)$.

$$\mathbf{r}_i^{(0)} = e_x(x_i) + e_y(y_i) \quad (i = 1, \dots, N), \quad (51.8)$$

$$\mathbf{s}_j^{(0)} = e_x(x_{\text{test}}^{(j)}) \quad (j = 1, \dots, n_{\text{te}}). \quad (51.9)$$

We do not explicitly mark the test points, as the model can infer that by seeing whether the bias vector of $e_y(\cdot)$ is present in the embedding. Stack our full state as a sequence $H^{(0)} = [\mathbf{r}_1^{(0)}, \dots, \mathbf{r}_N^{(0)}, \mathbf{s}_1^{(0)}, \dots, \mathbf{s}_{n_{\text{te}}}^{(0)}] \in \mathbb{R}^{(N+n_{\text{te}}) \times d}$.

The attention mask $A \in \mathbb{R}^{(N+n_{\text{te}}) \times (N+n_{\text{te}})}$ encodes the train invariance by letting all training samples attend to all training samples, and it encodes the independence of the test predictions by letting all test samples attend to all training samples and not to each other. The attention mask has the following form, where 0 means attention is possible, while $-\infty$ forbids attention

$$A_{uv} = \begin{cases} 0, & u \leq N, v \leq N \quad (\text{train} \leftrightarrow \text{train allowed}) \\ 0, & u > N, v \leq N \quad (\text{test} \rightarrow \text{train allowed}) \\ -\infty, & \text{otherwise} \quad (\text{no train} \rightarrow \text{test, no test} \leftrightarrow \text{test}). \end{cases}$$

For each layer $\ell = 1, \dots, L$ the architecture performs the standard transformer encoder operations for the given

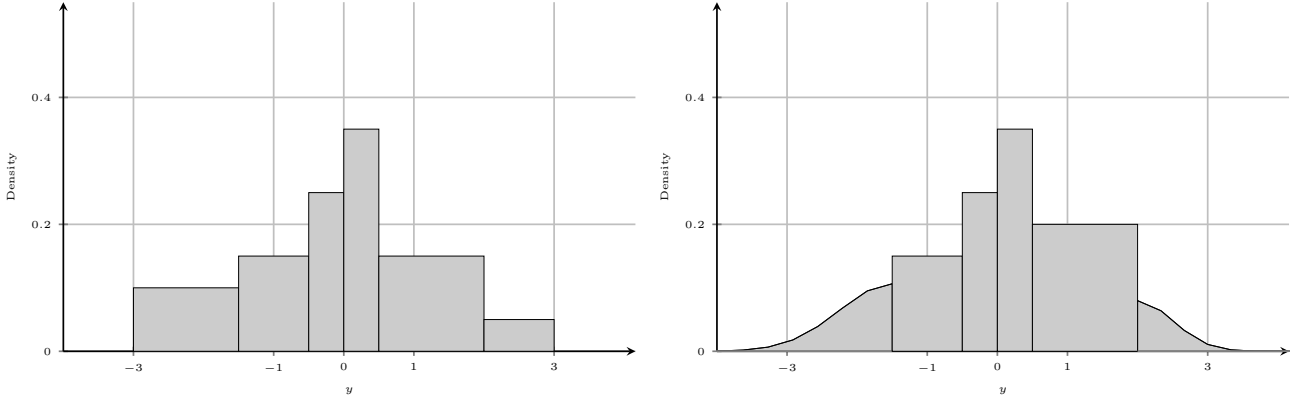


Figure 51.4: Two examples of a simple Riemann distribution with 7 borders (> 5000 borders are common in practice). On the left we see a distribution without full support, while it is extended to have full support on the right by employing half-normals on the outermost bars.

state, which are

$$Q_h = H^{(\ell-1)} W_{Q,h}^{(\ell)}, K_h = H^{(\ell-1)} W_{K,h}^{(\ell)}, V_h = H^{(\ell-1)} W_{V,h}^{(\ell)}, \quad (51.10)$$

$$\text{Attn}_h = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_h}} + A\right) V_h, \quad (51.11)$$

$$\tilde{H}^{(\ell)} = H^{(\ell-1)} + \text{LN}\left(\left[\text{Attn}_1 \parallel \dots \parallel \text{Attn}_H\right] W_O^{(\ell)}\right), \quad (51.12)$$

$$H^{(\ell)} = \tilde{H}^{(\ell)} + \text{LN}\left(\text{FFN}^{(\ell)}(\tilde{H}^{(\ell)})\right). \quad (51.13)$$

Finally, prediction heads are applied to the test states $\mathbf{s}_j^{(L)}$ to predict their respective y . For classification a standard softmax head is used, while curiously for regression a softmax is used as well. We will talk about this next.

A Versatile Output Distribution

PFNs that predict a scalar target y need to model the full distribution of $p(y|x, \mathcal{D})$ in a lot of use cases, necessitating a versatile output distribution. For this, PFNs typically use the so-called *Riemann distribution* [112, 1208], a continuous distribution with a piecewise constant density function, which discretises regression into a classification problem. It can be written as $q_\psi(y|x_{\text{test}}^{(j)}, \mathcal{D}) = \sum_{b=1}^B \pi_{j,b} \mathcal{U}([a_{b-1}, a_b])$, where there are $B+1$ boundaries with values a_b and $\pi_j = \text{softmax}(W_h \mathbf{s}_j^{(L)} + \mathbf{b}_h)$ is a standard softmax prediction. Its probability density function (PDF) looks like a histogram as we show in Figure 51.4 (left). The Riemann distribution is commonly used with half-normal distributions instead of the last block on each side, as that ensures the distribution's full support on $\mathbb{R}$. This is depicted in Figure 51.4 (right).

The Riemann distribution tends to require a significant amount of training data to attain high accuracy, but PFNs can be trained on virtually infinite data as the data is sampled from a prior. The upside of this distribution is that it allows the PFN to model a wide variety of posterior distributions, e.g., multimodal or asymmetrical distributions.

51.4.2 TabPFNv2 Architecture with 2-D Attention

Hollmann et al. [770] updated the PFN architecture introduced above to include attention across features, in addition to attention across samples. This type of architecture is becoming adopted quickly across tabular foundation models [408, 1861] and other applications like graph foundation models [495] and causal estimation [1414]. This updated architecture is further optimised for tabular tasks, where the inputs x are naturally modelled as a vector with permutation invariance. While previously every sample shared one state in the transformer, now every feature and every target is represented by a separate state, and we have one batched attention layer that attends across samples per feature as before, and one that attends across features per sample. The architecture is outlined in Figure 51.5.

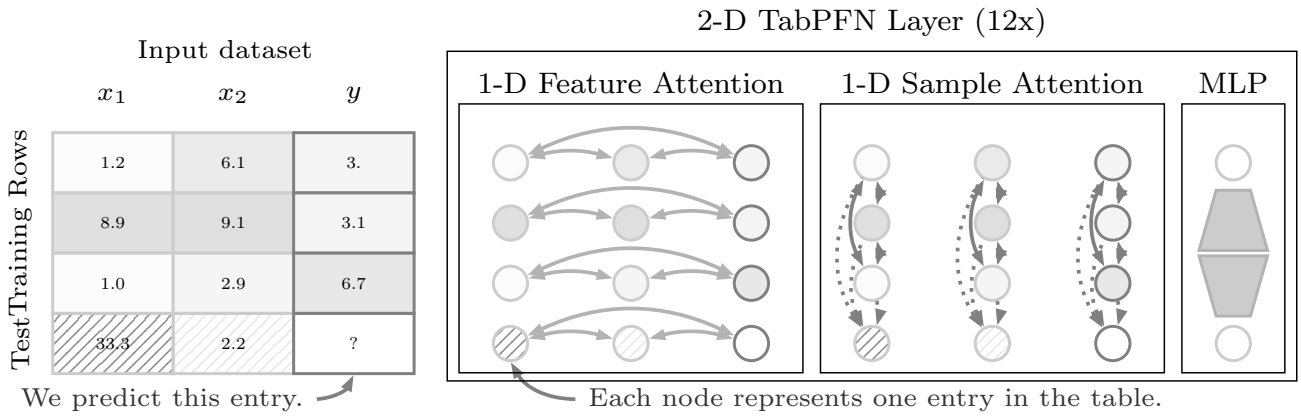


Figure 51.5: The architectural outline of the now commonly used TabPFNv2 architecture with 2-D attention.

51.5 Applications of Prior-data Fitted Networks

PFNs are used across a diverse set of domains – some PFN’s capabilities are very general, others narrow. So far, all of these domains share that the models work on featurised data, that is, x is a vector of meaningful entries. Since PFNs provide an approximation of the full PPD, they are also often used in settings where uncertainty estimation is important such as Bayesian optimisation. PFNs were so far only shown to work for smaller datasets of up to 50,000 points in context [1861], though. Let us review a set of applications.

51.5.1 Tabular Classification and Regression

In tabular prediction tasks, we are simply interested in using the PPD to label newly incoming datapoints for arbitrary featurised datasets. The TabPFN models [769, 770] popularised PFNs as foundation models for this application and strong descendants [1353, 1861] further established them in this domain. PFNs are, at the time of writing, seen as state-of-the-art on small to medium-sized tabular datasets (up to about 50,000 samples). They also tend to be much faster than previous methods, as they require only one or a few forward passes, rather than a whole training procedure encapsulated in a hyperparameter-tuning loop. These PFNs have priors that are carefully hand-crafted to represent tabular data well. The priors in this space are inspired by structural causal models, which take the flexible BNN example from Section 51.3.2 to an extreme. They allow for arbitrary sparsity, mixing of activation functions and noise distributions, mixing in of regression trees, as well as not restricting y to be the output and x to be the input of the network. These priors generate diverse synthetic tabular datasets, incorporating various types of feature relationships, noise patterns, and data distributions, that can also be seen in real-world data.

These models are a highlight of PFN development as they show how fast predictions can be with PFNs, while still performing very strongly on a very large and diverse domain. However, this domain might hit a ceiling of scaling PFNs to large in-context data, as posteriors eventually converge to a point estimate in the limit of infinite data, which renders Bayesian models less useful.

51.5.2 Bayesian Optimisation

As PFNs approximate the PPD, their output distribution can be leveraged for Bayesian optimisation (BO) [1209]. The goal of any BO algorithm is to maximise the output of an expensive-to-evaluate black-box function $f : \mathcal{X} \rightarrow \mathbb{R}$ with a limited number of function evaluations. The key component of any BO setup is a well-calibrated probabilistic model $p(f(x)|x, \mathcal{D})$, which PFNs provide. Its predictions are used to compute heuristics, so called acquisition functions, that give a trade-off between exploring new inputs x and exploiting regions, where f is known to return high values.

The classical probabilistic model used for BO is the GP. That is why in this domain it is most common to train PFNs on (generalisations of) GP priors. In this context, the advantages of PFNs compared to GPs lie especially in a wider choice of the prior, e.g., Müller et al. [1209] uses priors that involve ranges of kernel hyperparameters, input warping, and ignoring certain dimensions in a fully Bayesian way. Something that is very hard to achieve using MCMC sampling, which is usually employed for this purpose (see Chapter 1).

51.5.3 PFNs for SBI

Vetter et al. [1677] use the pre-trained TabPFNv2 to model the distribution over latent vectors in an SBI setup with a single fixed-size observation. That is, while PFNs can be seen as close kin to SBI, a pre-trained PFN can also be used to model the posterior in an SBI setting. The setup is that we have a prior $p(\theta)$ over latent vector θ and have a fixed size observation vector x , which we can sample via $p(x|\theta)$. The TabPFN is now used to model $p(\theta|x)$. While TabPFN is only trained to model univariate targets, Vetter et al. [1677] use multiple sequential TabPFN predictions to model $p(\theta|x)$ via the chain rule, predicting one entry after the other as $p(\theta_i|\theta_{<i}, x)$. TabPFN-based SBI showcases that while PFNs are trained on a similar objective compared to SBI, they typically target wider prior distributions, which, in this case entails some SBI applications.

51.5.4 Time Series Forecasting

Recent work has shown that PFNs can also be extended to the time domain, leaving the realm of permutation-invariant training and test sets. There are two successful lines of work to achieve this. i) One line of work [443, 1583] creates priors inspired by real-world time series to train PFNs on these with time being treated as a (specially encoded) feature in an architecture like the ones discussed above. ii) Another line of work uses pre-trained PFNs (TabPFN) to do forecasting, despite being trained only on tabular data [777]. This is achieved by converting time series data into a tabular format (e.g., using calendar features, lags).

51.5.5 Graphs

PFNs have very recently started to see some traction as a basis to train foundation models for node classification and regression. Similar to PFNs' value proposition for tabular data, they allow training a single foundation model that can perform tasks for which individual models used to be trained in a single forward-pass. There is work that does feature-engineering and fine-tuning on top of pre-trained tabular PFNs [496], but there is also work on building a prior for graphs specifically [495].

51.5.6 Causal Interventions

In tabular settings it is often of interest to understand the causal impact of changing an attribute on the outcome. In some cases the causal graph underlying a problem might be available, but in many cases it isn't. In these cases one previously needed to perform MCMC sampling over the very large, discrete set of possible causal graphs [80]. PFNs can amortise this effectively [80, 1099, 1414].

51.5.7 Learning Curve Extrapolation

PFNs have been applied to predict the future performance of machine learning models based on their training curves [6]. This is useful for hyperparameter optimisation: by predicting whether a configuration will ultimately achieve good performance, we can terminate unpromising runs early and allocate resources more efficiently. The PFN is trained on synthetic learning curves generated from a prior over training dynamics, enabling it to forecast performance with well-calibrated uncertainty.

51.5.8 Invariant Learning

PFNs, similar to Gaussian processes in many domains, allow adding invariances to models that might be necessary in some downstream tasks. This is explored for tabular prediction by Arbel et al. [44], who make classification PFNs equivariant to the order of the classes, and by Helli et al. [731], who make PFNs invariant to distribution shifts.

51.5.9 Aleatoric and Epistemic Uncertainty Decomposition

Similar to Gaussian processes, PFNs can be trained to predict both the noisy outcomes y and a non-noisy version f to differentiate aleatoric and epistemic uncertainty [125]. In Bergna et al. [125], the authors do this simply by training a PFN with two output heads one on the clean outcomes and one on outcomes with added noise. Similar to GPs, though, this simple approach only yields an approximation to the true decomposition and only is applicable to certain priors (like GP priors).

Chapter 52

Advances in singular learning theory for BDL

Bayesian deep learning is often presented as a clean marriage between Bayesian inference and deep neural networks. In practice, however, the mathematical picture many researchers carry over from standard statistical theory does not survive contact with the singular reality of deep learning. This mismatch leads to persistent misconceptions – not just about mathematical details, but about what quantities truly matter for understanding model complexity and predictive performance.

Singular learning theory (SLT), developed by Sumio Watanabe, is the asymptotic theory of Bayesian inference in singular models, i.e., models whose parameter–likelihood map is non-identifiable and/or Fisher information matrix is degenerate [1723]. Unlike in classical (regular) statistics – where each parameter corresponds to a unique distribution and Fisher information is positive definite – singular models have parameter symmetries, redundancies, and other complex geometric features which invalidate the standard quadratic approximation to the log-likelihood that underpins much of standard asymptotic statistical theory [1726, Ch. 1–2]. Such singular structures are pervasive in modern machine learning — from neural networks to mixture and latent-variable models — making SLT the natural asymptotic framework for their analysis.

SLT replaces the Gaussian asymptotics of regular models with an *algebraic–geometric* asymptotic description of the posterior [1723] controlled by the singularities of the Kullback–Leibler divergence between the true data-generating mechanism and the model, considered as function $K(w)$ of the model parameters. The key invariant of this KL divergence function is the **learning coefficient** λ , which controls the asymptotic behaviour of the volume of parameter space where K is small (see Equation (52.2) below). After a suitable reparametrisation—known as *resolution of singularities*— $K(w)$ locally factors into monomials and λ can be read off the exponents of these monomials [1726, Ch. 5, Def. 17; Ch. 6, Thm. 17] (Equation (52.6)). The learning coefficient can be regarded as a generalised parameter count that reduces to simple parameter counting in regular models. One important feature of the learning coefficient is that it depends on the true data-generating mechanism, and in which ways the model can approximate it.

The learning coefficient governs the first order behaviour of every major asymptotic property of Bayesian learning: the log marginal likelihood contains a $\lambda \log n$ term (Equation (52.10) below, [1726, Ch. 5, Thm. 11]), the Bayes generalisation error decays as λ/n (Equation (52.16), [1726, Ch. 5, Thm. 15]), and model selection criteria such as WAIC or WBIC provide valid extensions of AIC and BIC to singular models [1726, Ch. 8].

SLT is well suited to providing a rigorous language for describing the geometric degeneracies present in deep learning – permutation symmetries, scaling symmetries, and redundant units – that make standard Bayesian approximations fail. It tells us how Bayesian inference behaves when multiple parameter settings yield the same function, offering the only known asymptotic theory (in the large data limit $n \rightarrow \infty$) that remains correct in this regime¹. Besides the two textbooks [1723, 1726], we also recommend the surveys [1722, 1727].

In this chapter, we use SLT to clear up four common misconceptions in Bayesian deep learning:

Misconception 1: The Laplace approximation is an acceptable general-purpose asymptotic theory for Bayesian posteriors, even in singular models.

Misconception 2: The effective dimensionality of Bayesian neural networks is accurately and completely captured by the eigenspectrum of the Hessian.

¹In the limit where both $n, d \rightarrow \infty$ together, Bayesian asymptotic theory has also been studied for some specific models and data distributions, see e.g. [522, 1281]. What is perhaps most distinctive about SLT compared to these results is its generality.

Misconception 3: The generalisation error of the Bayes posterior predictive distribution is similar to that of the maximum a posteriori (MAP) estimator and the maximum likelihood estimator (MLE), asymptotically speaking, in singular models.

Misconception 4: Optimising the evidence lower bound (ELBO) in variational inference necessarily leads to good downstream predictive performance.

Behind all four misconceptions lurks a more basic objection: that singularities cannot matter in practice because real data never places a model exactly at a singularity. Section 52.2.1 addresses this objection directly: for a model that is merely *close* to singular, the singular description of Bayesian inference can be more accurate than the regular one over a very large range of sample sizes.

For each misconception, we will explain why the claim is false, present the mathematically correct statement, and illustrate its implications for Bayesian deep learning. Before doing so, we first give a primer on singular learning theory as it pertains to Bayesian deep learning.

52.1 Setup

Consider the problem of *parametric density estimation*: we are given data $\mathcal{D}_n = \{x_i\}_{i=1}^n$ sampled i.i.d. from a true data-generating distribution $p_0(x)$ on a sample space $X \subset \mathbb{R}^N$, we choose a (finite-dimensional) parametric model $p(x | w)$ of distributions on X indexed by a parameter $w \in \mathcal{W} \subset \mathbb{R}^d$, and we want to use this model to learn $p_0(x)$.

Superficially, this is quite different from (supervised) deep learning, where the model is a family of deterministic mappings $x \mapsto f_w(x)$ where w is the collection of neural network weights. However from a statistical and Bayesian perspective, we can augment such deterministic models by replacing $f_w(x)$ by a family of conditional distributions $p(y|x, w)$ with mean $f_w(x)$ and by modelling the input data distribution $p(x)$. This then turns Bayesian deep learning into a parametric density estimation problem in the sense above (on $X \times Y$). The deep linear network model in Section 52.2.2 is an example of this construction.

In the Bayesian formulation, uncertainty about the parameters is expressed through a prior distribution $\varphi(w)$ on w . The matter of eliciting a meaningful prior over neural network weights is nontrivial and forms an active research area; see Chapters 28–31 for a detailed treatment of priors in BDL. In this introductory chapter, we will work with a general prior $\varphi(w)$, which for simplicity of exposition we assume positive everywhere on $\mathcal{W}$.²

We will also assume that $p(x|w)$ and $\varphi(w)$ are **real analytic**. In theory, this assumption rules out neural networks that use ReLU activation functions. However, this is not a real obstacle as ReLU networks are still **piecewise analytic**, and are also known empirically to behave very similarly to related analytic activations like SiLU; we refer the reader to Wei et al. [1733] for further details. We will further assume the model is *well-specified* (or *realisable*, in the terminology of Watanabe), i.e., there exists $w_0 \in \mathcal{W}$ such that the true distribution $p_0(x)$ is realised as

$$p_0(x) = p(x | w_0).$$

This assumption deserves a comment, since it is in tension with Bayesian deep learning practice: for realistic data distributions and network architectures, no parameter reproduces $p_0(x)$ exactly, so practitioners essentially never operate in the realisable regime. We assume realisability purely for ease of exposition. The SLT results presented in this chapter do not require it: they extend to non-realisable settings in which the variance of the log-likelihood ratio process can be suitably controlled, the *relatively finite variance* condition of Watanabe [1726, Section 3.1, Definition 7]. We return to this point in Section 52.2.3, where realisability makes its only substantive appearance in this chapter: it is used to control the variance of the empirical process ξ_n underlying Watanabe’s asymptotics.

Define the *cross-entropy* between $p_0(\cdot)$ and $p(\cdot | w)$ (or *population negative log-likelihood*) as

$$L(w) := - \int p_0(x) \log p(x | w) dx,$$

and the closely related *Kullback–Leibler (KL) divergence* between $p_0(\cdot)$ and $p(\cdot | w)$ as

$$K(w) := \text{KL}(p_0(\cdot) \| p(\cdot | w)) = \int p_0(x) \log \frac{p_0(x)}{p(x | w)} dx,$$

²This assumption makes some of the statements cleaner, e.g. the learning coefficient λ becomes independent of the prior. The SLT framework does allow for more general priors; see [1723, Theorem 7.4] for how the learning coefficient changes when using the Jeffreys prior.

and write the set of KL minimisers as

$$W_0 := \{w \in \mathcal{W} : K(w) = 0\}.$$

Because we have assumed realisability, W_0 is non-empty. Note that the difference $L - K$ (the differential entropy of $p_0(x)$) is independent of w .

Regular vs. singular models. A model is *regular* if it is identifiable and the Fisher information matrix at each $w_0 \in W_0$ is positive definite; otherwise it is *singular*. In regular models, identifiability implies that W_0 consists of a single point. In singular models, by contrast, W_0 is a real analytic set which usually fails to be an analytic manifold due to the presence of singularities [1723]. See Section 52.2 below for some simple examples of W_0 .

Posterior distribution. Given data $\mathcal{D}_n = \{x_i\}_{i=1}^n$, the *empirical negative log-likelihood* is

$$L_n(w) = -\frac{1}{n} \sum_{i=1}^n \log p(x_i | w).$$

Most machine learning loss functions (mean-squared error, cross-entropy) can be written as L_n , in which case it might be called a training loss. The posterior distribution can be written in terms of L_n as

$$\pi(w | \mathcal{D}_n) \propto \exp\{-nL_n(w)\} \varphi(w) dw,$$

with normalising constant

$$Z_n = \int_{\mathcal{W}} e^{-nL_n(w)} \varphi(w) dw,$$

which is also known as the marginal likelihood. Related to the empirical negative log-likelihood is the empirical KL divergence

$$K_n(w) := \frac{1}{n} \sum_{i=1}^n \log \frac{p_0(x_i)}{p(x_i | w)},$$

with which we can alternatively write the posterior as

$$\pi(w | \mathcal{D}_n) \propto \exp\{-nK_n(w)\} \varphi(w) dw.$$

Let us call the corresponding normalising constant the *normalised marginal likelihood*

$$Z_n^0 = \int_{\mathcal{W}} e^{-nK_n(w)} \varphi(w) dw.$$

Note that the difference $K_n - L_n$ as well as the quotient Z_n^0/Z_n are independent of w (and related to the entropy of the data distribution). Moreover, treating the data as a random sample from $p_0(x)$, we have $L = \mathbb{E}_{p_0}[L_n]$ and $K = \mathbb{E}_{p_0}[K_n]$; in machine learning terminology, this means that L is the *population loss function*.

Posterior predictive distribution. Rather than relying on point estimates such as the MLE or MAP, Bayesian deep learning makes predictions by averaging over parameter uncertainty via the *posterior predictive distribution* (PPD)

$$\hat{p}_{\text{Bayes}}(x) := \int_{\mathcal{W}} p(x | w) \pi(w | \mathcal{D}_n) dw. \quad (52.1)$$

In regular models, where the posterior contracts to a point mass, the Bayes PPD behaves asymptotically like the MLE and MAP predictors [1726, Ch. 4.5]. In singular models, however, the posterior does not usually collapse to a single point, but rather concentrates around higher-dimensional subsets of W_0 , and singular learning theory shows that the Bayes PPD achieves different expected generalisation error asymptotically than either MLE or MAP [1723, Ch. 6.4]; we will come back to this point in Section 52.5.

Now we come to a very important but subtle point. By the realisability assumption, the models $p(x|w_0)$ for all $w_0 \in W_0$ are all equal to the same (true) distribution. In the supervised context, this means that all models in W_0 compute the same function $f_0(x)$. However, they are *not* equal in the eyes of the Bayesian posterior! On reflection, this is quite intuitive because the structure of the model *around* different elements of W_0 can be quite different, and the Bayesian posterior is sensitive to this. This intuition predicts that the posterior distribution concentrates around a specific subset of “maximally singular” parameters $w_0 \in W_0$, those for which the model is minimally sensitive to small changes of parameters. SLT turns this vague picture into solid mathematics.

52.2 Deep learning is singular

Neural networks (deep or shallow) are *inherently* singular, for several overlapping reasons [1733]. First, there are global symmetries of the model, both discrete and continuous (see Chapter 32 for a systematic treatment of symmetries in Bayesian neural networks):

- **Permutation symmetries.** In a fully connected network with hidden units, permuting the units within a layer (and permuting the corresponding rows/columns of weight matrices and biases) leaves the input–output map unchanged. These permutations create large equivalence classes in $\mathcal{W}$ mapping to the same model output, so W_0 is not a singleton and therefore the probabilistic neural network is not identifiable.
- **Scaling symmetries.** For activation functions such as ReLU, scaling the incoming weights to a hidden unit by a constant c and scaling its outgoing weights by $1/c$ leaves the network output unchanged. This introduces continuous curves of equivalent parameters in W_0 , again leading to non-identifiability.
- **Linear symmetries.** Whenever a neural network architecture involves a product AB of weight matrices, the transformation $(A, B) \mapsto (AP^{-1}, PB)$ with P an invertible matrix does not change the input–output map. This happens for example with the query and key matrices in the attention mechanism of transformers, and also for the toy model of deep linear networks (which we discuss briefly at the end of Section 52.2.2).
- **Batch and layer-norm.** Applying batch- and/or layer-normalisation makes the input–output map invariant under certain affine transformations of the weights.

Global continuous symmetries of the model have been studied extensively in Bayesian and geometric deep learning. For example, such symmetries force the Hessian of the empirical loss function L_n to be singular (have exactly vanishing eigenvalues) at every critical point, in particular everywhere on W_0 . As such, these models fall outside the class of regular models and understanding their Bayesian learning theory requires SLT.

Besides these global (or *generic*) singularities, which are always present independently of the true data-generating mechanism $p_0(x)$, singularities can also arise from the relationship between $p_0(\cdot)$ and the model $p(\cdot|w)$, specifically, from the geometry of W_0 and of the model around parameters $w_0 \in W_0$. For concreteness, we will give examples in the context of neural networks, but such local singularities occur in essentially any parametric model with hierarchical structure, mixtures, latent variables, etc.

- **Redundant units.** Suppose that a network $f_{w_0}(x)$ for some $w_0 \in \mathcal{W}$ has a unit with zero incoming weights. Then the choice of outgoing weights does not change the input–output mapping. Geometrically, this means that W_0 contains a large submanifold around w_0 , creating flat directions in the loss landscape. Similarly, suppose that $f_{w_0}(x)$ has two units with identical incoming weights. Then exchanging these units is an additional (discrete) symmetry.
- **Dead and saturated units** For ReLU networks, suppose that a particular ReLU unit never activates on the input data distribution (for example because its bias is very negative), then both incoming and outgoing weights can be changed in a neighbourhood of w_0 (up until the point that they change the preactivation distribution enough to activate the ReLU). Conversely, ReLU units that are always on behave like linear activations, and this leads to additional local linear symmetries of the type discussed above.

Because of such phenomena, the geometry of W_0 and the type of singularities depends on the true data-generating distribution $p_0(x)$. This turns out to have drastic consequences on asymptotic Bayesian inference, which are precisely quantified by SLT.

A simple example. Consider a two-layer, or one-hidden-layer, ReLU network

$$f_w(x) = v^\top \sigma(Ux),$$

with $U \in \mathbb{R}^{m \times p}$, $v \in \mathbb{R}^m$, and $\sigma(z) = \max(z, 0)$ applied elementwise. If $w = (U, v)$ is a solution with $K(U, v) = 0$, then so is (PU, Pv) for any permutation matrix P (unit permutation symmetry), and so is $(DU, D^{-1}v)$ for any invertible diagonal matrix D with positive entries (scaling symmetry). These transformations generate continuous and discrete families of equivalent parameters, and they will engender complex geometry in W_0 .

Singularities and model complexity Note that, in the above examples, local singularities emerge because the distribution $p_0(x)$ is *special* or *simple* compared to the distribution $p(x|w)$ for a generic choice of $w \in \mathcal{W}$. Moreover, those examples are not the only phenomena that lead to such local singularities; there can be more subtle interactions between $p_0(x)$ and how it can be represented by the model.

This theoretical discussion is connected to an important empirical aspect of deep learning. Indeed, neural networks trained on “natural” data distributions seem to be able to learn them much more efficiently and with less overfitting than predicted by regular statistical learning theory given their parameter count. From the singular Bayesian perspective, this precisely suggests that neural networks and natural data distributions are special in the sense that trained models (in W_0) have many additional local singularities.

52.2.1 Do singularities really matter?

A common misconception about SLT is that singularities, especially local singularities which depend on the data-generating distribution $p_0(x)$, are irrelevant because they are never exactly realised in practice.

The idea has a grain of truth to it; for a given model, the set of parameters in $\mathcal{W}$ which exhibit additional singularities (besides the generic ones coming from architectural symmetries) form a set of measure 0, and almost every small perturbation of $p_0(x)$ leads to a situation with only generic symmetries. While models with generic symmetries are still singular and one cannot apply regular statistical theory to them, there are sometimes simple transformations of the model which “mod out” those symmetries and reduce to the regular case.

However, we have to remember that both regular statistical learning theory and SLT are *asymptotic* descriptions of learning as $n \rightarrow \infty$. When the model $(p(x|w), p_0(x))$ is regular but $p_0(x)$ is very close to $p'_0(x)$ such that $(p(x|w), p'_0(x))$ is very singular, SLT applied to the $(p(x|w), p'_0(x))$ can be a more accurate description of Bayesian inference than the regular theory applied to $(p(x|w), p_0(x))$ for a possibly very large range of n (which can very much include the size of the dataset we actually care about, if that data is insufficient to distinguish between $p_0(x)$ and $p'_0(x)$!). In other words, for an “almost singular” model, you may need a *lot* of data for asymptotic normality of the posterior to kick in! See [1726, Figure 1.5] for an illustration of this in a simple mixture model.

Takeaways:

Because of these built-in symmetries and redundancies, deep networks *always* violate the regularity conditions used in classical asymptotic theory. Moreover, they interact in a complicated way with the precise structure of the training data distribution, and different solutions in W_0 can have different generalisation behaviour on new data. Deep learning therefore fits especially well with the framework of SLT, which is precisely designed to tackle those phenomena.

52.2.2 The learning coefficient

In regular models, a quadratic (Laplace) expansion of the log-likelihood around the true parameter yields Gaussian asymptotics of the posterior distribution as $n \rightarrow \infty$ (the celebrated Bernstein–von Mises theorem). In singular models, the posterior is not asymptotically normal and one cannot apply these standard results.

One fundamental insight of SLT is that there is still an asymptotic representation which is controlled by the singularities of $K(w)$. The precise form of this representation is complicated and we will only briefly discuss it in Section 52.2.3. However, its main downstream statistical effects are controlled by a single number, the *learning coefficient* or *real log-canonical threshold*³ (RLCT), $\lambda = \lambda(K) > 0$, which we now introduce.

Volume of near-optimal parameters. For a small tolerance $\varepsilon > 0$, consider the prior-weighted volume

$$\text{Vol}(\varepsilon) := \int_{K(w) < \varepsilon} \varphi(w) dw$$

of parameter values that achieve KL divergence no larger than ε . Then the RLCT is defined as the constant λ such that [1723, Thm. 7.1]

$$\text{Vol}(\varepsilon) \sim C \varepsilon^\lambda (-\log(\varepsilon))^{m-1} \quad \text{as } \varepsilon \rightarrow 0, \quad (52.2)$$

for some constant $C > 0$ and some integer⁴ $m \in \mathbb{N}_{\geq 1}$. The asymptotic (52.2) determines λ through a ratio in which both the constant C and the logarithmic correction cancel: for any $a > 1$,

$$\frac{\text{Vol}(a\varepsilon)}{\text{Vol}(\varepsilon)} \sim a^\lambda \left(\frac{-\log(\varepsilon) - \log(a)}{-\log(\varepsilon)} \right)^{m-1} \longrightarrow a^\lambda \quad \text{as } \varepsilon \rightarrow 0,$$

³The name “real log-canonical threshold” originates from singularity theory in algebraic and analytic geometry (we will not attempt to explain here what is “log-canonical” about it!). We usually use “real log-canonical threshold” when we consider λ as a geometric invariant of the function K , and “learning coefficient” when we emphasise its statistical significance.

⁴This (*real log-canonical*) *multiplicity* is a secondary geometric invariant of the singularities which we will largely ignore in this brief survey.

since the ratio of the logarithmic factors tends to 1. Taking logarithms yields

$$\lambda = \lim_{\varepsilon \rightarrow 0} \frac{1}{\log(a)} \log \left(\frac{\text{Vol}(a\varepsilon)}{\text{Vol}(\varepsilon)} \right). \quad (52.3)$$

In this definition, K can be any nonnegative real analytic function with $K^{-1}(0) \neq \emptyset$. It is then easy to see that λ is also the supremum of all $s > 0$ such that the function K^{-s} is locally integrable.

Regular and singular models. Recall that for regular models, $K(w)$ is locally a positive definite quadratic form (Morse lemma) so the set $\{K(w) < \varepsilon\}$ is an ellipsoid of radius $\propto \sqrt{\varepsilon}$. Combining Equation (52.2) with the formula for volumes of spheres (or, alternatively, combining the generalised Laplace asymptotic (52.4) of Section 52.2.3 with the standard Laplace approximation), we see that $\lambda = d/2$ for regular models.

Conversely, for any singular model, $K(w)$ can vanish along curves, surfaces, or other manifolds within $\mathcal{W}$. In fact, we then always have $\lambda < d/2$. Geometrically, λ quantifies how rapidly the set of nearly optimal parameters contracts as we tighten the accuracy requirement around the true distribution. Large values of λ correspond to sharp, isolated optima, while small values signal wide or degenerate valleys in parameter space where many different w yield nearly identical predictions. However, despite some apparent similarities, λ is of a different nature than curvature/flatness measures based e.g. on the Hessian eigenspectrum. In particular, λ is invariant under analytic changes of coordinates (bianalytic maps).

Deriving λ in closed-form. There are a few simple functions K where it is easy to derive λ . For example, $\lambda(u^{2k}) = \frac{1}{2k}$ and more generally $\lambda(u_1^{2k_1} u_2^{2k_2} \dots u_d^{2k_d}) = \min \left\{ \frac{1}{2k_i} \right\}$. Beyond those simple cases, determining exact values of λ is challenging.

First off, λ depends on the true distribution $p_0(x)$, which we do not have access to. Moreover, even in theoretical settings where access to $p_0(x)$ is assumed, computing λ is in general a difficult algebraic geometry question. This has nevertheless been achieved in a number of models; see [1727, End of Section 3] for a list of references. The most relevant computations for Bayesian deep learning so far are due to Aoyagi for fully connected deep linear networks [39] and for one-hidden-layer ReLU networks [40]. We briefly summarise the linear result below.

Learning coefficient of deep linear networks. Consider a fully connected deep linear network with L hidden layers, input dimension $d_0 = d_{in}$, output dimension $d_{L+1} = d_{out}$ and hidden widths d_i for $0 < i < L + 1$. The weight space $\mathcal{W}$ consists of tuples of matrices $A_\bullet = (A_0, \dots, A_L)$ with $A_i \in \mathbb{R}^{d_{i+1} \times d_i}$ for $i = 0, \dots, L$. Note that the total parameter count of the model is $D = \sum_i d_i d_{i+1}$. The function computed by the network is simply the product matrix $A_L \dots A_0$. We turn this into a probabilistic model as

$$p(y, x | A_\bullet) \propto \exp \left(-\frac{1}{2} \|y - A_L \dots A_0 x\|^2 \right) p(x)$$

with the input distribution $p(x)$ also isotropic Gaussian. We also assume the true distribution is realisable: $p_0(y, x) = p(y, x | A_\bullet^*)$ for some $A_\bullet^*$. Then one computes that

$$K(A_\bullet) \propto \int \| (A_L \dots A_0 - A_L^* \dots A_0^*) x \|^2 p(x) dx.$$

Let $r := \text{rank}(A_L^* \dots A_0^*) \leq \min d_i$. The model and the function K have a large group of continuous linear symmetries (which becomes larger as r decreases), which can be used to study the geometry of W_0 . Aoyagi gives a formula for $\lambda = \lambda(K)$ in terms of r and the widths d_i [39, Theorem 1]. The formula is combinatorially involved and we state two representative special cases. First for a linear network of constant width ($d_i = d$) and sufficiently deep ($L > d - r$):

$$\lambda = \frac{1}{4}(d^2 + d + r^2 - r).$$

Then, for a very wide network ($d_i > d_{in} + d_{out} - r$ for all $0 < i < L + 1$):

$$\lambda = \frac{1}{2} d_{in} d_{out}.$$

To get a feel for the constant-width formula, note that for a full-rank target ($r = d$) it gives $\lambda = d^2/2$, already much smaller than $D/2 = (L + 1)d^2/2$. The gap widens for rank-deficient targets: with $d = 10$ and $r = 5$, the condition $L > d - r$ permits $L = 6$, and the formula gives $\lambda = \frac{1}{4}(100 + 10 + 25 - 5) = 32.5$, compared with $D/2 = 350$. Note that in both cases, λ is much smaller than $D/2$ and does not depend on the depth L ; this is

true in general as long as L is sufficiently large. In the first case, we also observe that λ is an increasing function of r ; more generally, λ is always non-decreasing in r . This shows that deep linear networks are very singular, and become more so as the depth increases or the rank of the target function decreases.

Estimating λ . Given the difficulty of exact computation of λ , we can ask about how to *estimate* it from data. SLT provides tools to this effect, the *WBIC* in particular, see Section 52.3. The learning coefficient estimator based on the WBIC is however quite unstable and in our own research we never had much success in getting good estimates. This is perhaps not too surprising, since sampling from the full posterior distribution is challenging. In Section 52.4.1 we introduce a quantity related to the learning coefficient that we call the **local learning coefficient (LLC)**, as well as the *LLC estimator*. The LLC is both easier to estimate numerically and more sensitive to the local structure of the model, and for those reasons has seen significant uptake since it was first introduced in Lau et al. [978].

52.2.3 Resolution of singularities and mixture of standard forms

From the Bayesian perspective, the main property of λ is that it controls the asymptotics of *exponential integrals* of the KL-divergence K , generalising the standard Laplace approximation. More precisely⁵, we have

$$\int_{\mathcal{W}} e^{-nK(w)} \varphi(w) dw \underset{n \rightarrow \infty}{\propto} n^{-\lambda} \log(n)^{m-1}. \quad (52.4)$$

Note the (unnormalised) posterior distribution is obtained by replacing K by the K_n in the LHS of (52.4), and that λ does not depend on the (positive) prior φ . One of Watanabe’s main technical achievements is to combine the algebro-geometric techniques underlying this deterministic asymptotic result with probabilistic techniques from *empirical process theory* to derive a precise asymptotic description of the posterior itself. We briefly review these techniques below.

Standard form. By applying a change of variables known in algebraic geometry as a *resolution of singularities*⁶, one can introduce a finite collection of analytic charts $g_j : U_j \rightarrow \mathcal{W}$, $j = 1, \dots, J$, such that the parameter regions near $W_0 = \{w : K(w) = 0\}$ are covered by their images, and local coordinate systems $u = (u_1, \dots, u_d)$ on each U_j such that the Kullback–Leibler divergence takes the *standard (or monomial) form*

$$K(g_j(u)) = u^{2k^{(j)}} = u_1^{2k_1^{(j)}} \cdots u_d^{2k_d^{(j)}}, \quad (52.5)$$

where $k^{(j)} = (k_1^{(j)}, \dots, k_d^{(j)})$ is a multi-index of nonnegative integers [1723, Thm. 6.5].

Geometric interpretation. The collection of charts $\{(U_j, g_j)\}$ thus provides a local description which “untangles” the singularities of the model near W_0 . The coordinates $u = (u_1, \dots, u_d)$ separate degenerate directions in the original space $\mathcal{W}$ (creating ridges, folds, or intersections within W_0) from the active ones that determine the local asymptotic behaviour of the posterior: coordinates with $k_i^{(j)} > 1$ (resp. $k_i^{(j)} = 0$) represent flatter (resp. completely flat) directions.

Posterior vs generalised Laplace integral. While $K_n \neq K$, we have $\mathbb{E}[K_n] = K$, and Watanabe uses empirical process theory⁷ to write

$$K_n(g_j(u)) = K(g_j(u)) - \sqrt{\frac{K(g_j(u))}{n}} \xi_n(u)$$

where $\xi_n(u)$ converges in law to an explicit Gaussian process $\xi(u)$. This more or less reduces the study of the posterior to two contributions: the deterministic generalised Laplace integral in the LHS of Equation (52.4) and the process⁸ ξ .

Mixture of standard forms. Substituting the standard form representation in each local chart gives

$$\int_{\mathcal{W}} e^{-nK(w)} \varphi(w) dw \asymp \sum_{j=1}^J \int_{U_j} e^{-nK(g_j(u))} \varphi(g_j(u)) |g'_j(u)| du,$$

⁵Still assuming for simplicity that the model is well-specified ($W_0 \neq \emptyset$), that $\varphi > 0$, and under some minor additional convergence assumptions which we will not spell out here.

⁶We refer to [1723, §3.5] for a down-to-earth introduction to resolution of singularities and Hironaka’s theorem. We note in passing that a function K admits many resolutions and that there is no “canonical” choice.

⁷It is precisely for this step that the well-specified assumption comes into play, to bound the *variance* of K_n around K in order to properly apply the CLT. For non well-specified models, everything goes through as long as this variance can be controlled. See [1726, Section 3.1, Definition 7] for a sufficient condition which allows the theory to go through with only minor changes.

⁸The effect of ξ is secondary and we mostly neglect it in this chapter.

so that each term corresponds to the contribution from one local chart [1726, Ch. 6.3]. The analytic map g_j can also be chosen to factorise the prior and Jacobian determinant as

$$\varphi(g_j(u))|g'_j(u)| = b_j(u) |u^{h^{(j)}}| = b_j(u) \left| u_1^{h_1^{(j)}} \cdots u_d^{h_d^{(j)}} \right|,$$

where $h^{(j)} = (h_1^{(j)}, \dots, h_d^{(j)})$ is a multi-index of nonnegative integers and $b_j(u)$ is analytic and strictly positive near the origin [1723, Thm. 6.5]. The integers $h_i^{(j)}$ describe the order of vanishing of the Jacobian along each coordinate axis, so that the change of variables locally scales volume by a monomial factor $|u^{h^{(j)}}|$. By a simple computation for monomials, we now obtain [1723, Thm. 6.7]

$$\int_{U_j} e^{-nK(g_j(u))} \varphi(g_j(u)) |g'_j(u)| du = C_j n^{-\lambda_j} (\log n)^{m_j-1} (1 + o(1)),$$

with

$$\lambda_j = \min_{1 \leq i \leq d} \frac{h_i^{(j)} + 1}{2k_i^{(j)}}, \quad m_j = \# \left\{ i : \frac{h_i^{(j)} + 1}{2k_i^{(j)}} = \lambda_j \right\},$$

so λ_j captures the dominant rate of decay in a given chart and m_j the number of coordinates that achieve it [1726, Ch. 5, Def. 17].

Consequently, for large n , and after passing to the resolution, the posterior takes the form of a finite mixture of “standard-form” components, each characterised by its own local geometry (λ_j, m_j) . Combining with Equation (52.4), we obtain that the **global** learning coefficient λ and multiplicity m introduced in Section 52.2.2 is given by

$$\lambda = \min_{1 \leq j \leq J} \lambda_j, \quad m = \max_{1 \leq j \leq J} \# \{ i : \lambda = \frac{h_i^{(j)} + 1}{2k_i^{(j)}} \}. \quad (52.6)$$

Note that in regular models, $K(w) \approx w_1^2 + \dots + w_d^2$ is not in standard form when $d > 1$. The resolution of singularities in this case is very explicit (the “blow-up” of $\mathcal{W}$ at the unique point in W_0) and described (in dimension 2, but the general case is similar) in [1723, Example 3.15].

52.3 Misconception 1: Laplace approximation is valid for singular models

We now come to addressing the first misconception, that the Laplace approximation is appropriate for singular models. The discussion from the previous sections has already foreshadowed this, but we will now dig into the details. The Laplace approximation is used in two distinct but related ways in Bayesian deep learning: to approximate the posterior distribution for downstream prediction and to estimate the model evidence for model selection. We will address the invalidity of the Laplace approximation in each use case and point to the SLT corrective, where possible.

52.3.1 Laplace for prediction

As discussed in Section 52.2.3, the Laplace approximation cannot in general describe the posterior in singular models, including neural networks. After a resolution of singularities, the posterior decomposes into a mixture of finitely many local components, each governed by its own pair (λ_j, m_j) . This asymptotic structure is fundamentally incompatible with the single Gaussian form assumed by the Laplace method—or even with a finite mixture of Gaussians.

In regular models, by contrast, the Laplace approximation is mathematically justified: for large n , the posterior is approximately Gaussian,

$$\pi(w|\mathcal{D}_n) \approx \mathcal{N}(w_{\text{MAP}}, H(w_{\text{MAP}})^{-1}), \quad (52.7)$$

where w_{MAP} is the maximum a posteriori (MAP) estimate and H is the **Hessian of the negative log posterior** evaluated at w_{MAP} .

This Gaussian approximation is widely used in Bayesian deep learning to approximate the PPD, as it allows efficient sampling. The typical procedure is:

1. Train a neural network using stochastic optimisation and take the resulting weights $\hat{w}$ as w_{MAP} ;

2. Sample weights from $\mathcal{N}(\hat{w}, H(\hat{w})^{-1})$ to perform Monte Carlo integration for the PPD.

However, in deep networks, $H(\hat{w})$, the Hessian at a trained solution, $\hat{w}$, is typically singular or nearly so, making the Gaussian ill-defined. Practical implementations therefore regularise H —for instance by adding damping or diagonal corrections—yielding numerically stable but ad hoc covariance structures. Moreover even after such corrections computing H is very expensive and is usually replaced by more tractable approximations such as Generalised Gauss–Newton (GGN) or Kronecker-Factored Approximate Curvature (K-FAC); see Chapter 40 for a detailed treatment of scalable Laplace approximations. These modifications produce Laplace-inspired approximations: they retain the computational form of the Laplace method but no longer correspond to the true local curvature of the posterior.

Despite these theoretical shortcomings, Laplace-based methods can perform well empirically, though the evidence is specific to the task. Because they perturb around a trained MAP estimate, they essentially inherit its in-distribution accuracy, and they typically improve calibration on in-distribution data [380]. This success also depends on further engineering choices: sampling naively from a full-network Laplace posterior underfits severely, and practical implementations rely on the linearised predictive [813] (Chapter 8), last-layer application, and prior precision tuned via the marginal likelihood. The quality of uncertainty quantification away from the training distribution is less settled: diagonal approximations fail to represent uncertainty between clusters of training data [542] and Gaussian posterior approximations degrade under dataset shift [1274], although even a last-layer Gaussian provably mitigates the overconfidence of ReLU networks far from the data [928]. None of this implies that the Laplace approximation is valid for neural networks; it reflects the interplay between optimisation dynamics, curvature approximations, and regularisation choices that may yield effective but heuristic posterior samplers. In other words, although SLT clarifies why the Laplace approximation does not hold in singular models, it does not quite tell us why (and when) its variants prove useful in practice. In addition SLT does not directly offer a computable alternative for approximating the PPD. Nonetheless it provides a geometric framework that identifies what the Laplace approximation overlooks—and therefore, what a successful approximation needs to capture. One example, developed in Section 52.5, concerns prediction: a single Gaussian centred at one point of W_0 misses the posterior mass spread across the rest of W_0 , and averaging over W_0 is precisely the source of the asymptotic predictive advantage of the exact Bayes PPD. The good in-distribution performance of Laplace variants is therefore performance at the level of point estimates, and is fully consistent with the marginalisation gains discussed there.

52.3.2 Laplace for model selection

The second key use of the Laplace approximation in Bayesian deep learning is to estimate the marginal likelihood (or model evidence), $Z_n = \int_{\mathcal{W}} e^{-nL_n(w)} \varphi(w) dw$, or equivalently the log marginal likelihood (LML), $\log Z_n$. The LML can be used to select hyperparameters, such as the prior precision (weight decay), a process known as Type-II maximum likelihood or empirical Bayes. The Laplace approximation provides a tractable expression for the LML in regular models:

$$\log Z_n \approx -nL_n(w_{\text{MAP}}) - \frac{d}{2} \log(n) + \log \varphi(w_{\text{MAP}}) + \frac{d}{2} \log(2\pi) - \frac{1}{2} \log \det(\bar{H}(w_{\text{MAP}})) \quad (52.8)$$

where $\bar{H}(w_{\text{MAP}}) := H(w_{\text{MAP}})/n$ is the Hessian of the negative log-posterior from Equation (52.7) rescaled by the sample size [1105]; for large n , $\bar{H}$ is approximately the Hessian of the empirical negative log-likelihood L_n .

Asymptotic expansion for singular models. SLT shows that the approximation (52.8) of the LML is incorrect for singular models. Indeed, a direct application of Watanabe’s asymptotic expansion of the (unnormalised) posterior discussed in Section 52.2.3 leads to

$$\log Z_n = -nL_n(w_0) - \lambda \log n + (m-1) \log \log n - F_n^R, \quad (52.9)$$

where w_0 is any parameter in W_0 and F_n^R is a random variable which converges in distribution to a constant ([1723, Corollary 6.1]; compare this with the closely related Equation (52.4)). Because w_0 is not known, we can consider plugging in the MAP estimator⁹, leading to

$$\log Z_n \approx -nL_n(w_{\text{MAP}}) - \lambda \log n + (m-1) \log \log n + O_p(1). \quad (52.10)$$

This expression generalises the Laplace approximation (52.8), since we have $\lambda = d/2$ and $m = 1$ for regular models. Note that, like in the regular case, the constant order term in Equation (52.10) is the first term in this asymptotic expansion which depends on the prior φ ; however, for singular models, that constant order term is

⁹This is valid when for instance the likelihood ratio between w_0 and w_{MAP} is bounded in probability.

more complicated and does not (only) depend on the prior and the Hessian but on the precise geometry of the model (through the Gaussian process ξ mentioned in Section 52.2.3).

Estimating the LML in practice: WBIC. The equations (52.9) and (52.10) are useful to understand the theoretical role of λ in model selection. However to make use of them to approximate the log marginal likelihood, we a priori need to know λ (which itself depends on $p_0(x)$). Using the same proof techniques, Watanabe derived an estimator of the LML which does not require λ , the **Widely Applicable Bayesian Information Criterion (WBIC)** [1725]:

$$\text{WBIC}_n = \mathbb{E}_w^\beta[nL_n(w)],$$

for $\beta = 1/(\log n)$ where $\mathbb{E}_w^\beta$ is the expectation with respect to the *tempered* Bayesian posterior with *inverse temperature* β :

$$p^\beta(w|\mathcal{D}_n) \propto \exp\{-n\beta L_n(w)\}\varphi(w).$$

Watanabe showed $\text{WBIC}_n = -\log Z_n + O(\sqrt{\log n})$ so that

$$\text{WBIC}_n = nL_n(w_0) + \lambda \log n + O(\sqrt{\log n}). \quad (52.11)$$

Estimating the learning coefficient using the WBIC. We end this section by briefly noting that the WBIC estimator can be turned around into an estimator of the learning coefficient itself. Let $\beta_1 = \beta_{01}/\log n$ and $\beta_2 = \beta_{02}/\log n$ where β_{01} and β_{02} are positive constants. Then we have that the following is a consistent estimator of the learning coefficient

$$\hat{\lambda} = \frac{\mathbb{E}_w^{\beta_1}[nL_n(w)] - \mathbb{E}_w^{\beta_2}[nL_n(w)]}{1/\beta_1 - 1/\beta_2}.$$

52.4 Misconception 2: Effective dimensionality is captured by the eigen-spectrum of the Hessian

Classical learning theory suggests that models with far more parameters than data points should overfit catastrophically, yet in deep learning practice, larger models often generalise even better. This paradox, which has been discussed at length in Bayesian deep learning (including MacKay [1104], Maddox et al. [1117], Wilson [1768]) reveals a fundamental flaw in relying on simple parameter counting as a measure of model complexity.

To resolve this paradox, one needs alternative complexity measures that are not based on the raw parameter count, but on the model's interaction with the data. One approach is to consider the eigenspectrum of the Hessian of the training loss at a trained model, $H(\hat{w}) := \nabla^2 L_n(\hat{w})$. Empirically, H typically admits a few large positive eigenvalues, a long tail of small positive eigenvalues, some exactly vanishing eigenvalues induced by architectural symmetries, and some small negative eigenvalues. From the perspective of the Laplace approximation of the posterior, one can hope that the number of “large” positive eigenvalues (the “*effective rank*” of H) quantifies posterior contraction—essentially, how many parameter directions are meaningfully constrained by the training data and truly “used” by the model. Consequently, when comparing models that have achieved a similarly low training loss, one could expect that the one with a lower effective Hessian rank generalises better.

There are various ways to operationalise the effective rank of H . A specific implementation is the **effective dimensionality**, denoted N_{eff} . It originated with MacKay [1104], was later modernised by Maddox et al. [1117] (who positioned it as a generalisation metric for deep learning) and further developed by Wilson [1768] (who used it to argue that large models succeed by learning a compressed representation). The effective dimensionality is defined as

$$N_{\text{eff}}(H, z) := \sum_{i=1}^d \frac{\mu_i}{\mu_i + z}, \quad (52.12)$$

where μ_i are the eigenvalues of H and $z > 0$ is a regularisation constant. In this formula, eigenvalues $\mu_i \gg z$ contribute approximately 1 to the sum, while small eigenvalues contribute approximately 0, so N_{eff} can indeed be interpreted as an effective rank of H . Although Hessians of neural networks commonly have negative eigenvalues, Maddox et al. [1117] suggest they are largely irrelevant for the calculation, as their magnitudes are typically small compared to the large positive eigenvalues. In line with the discussion above, Maddox et al. [1117] demonstrate that N_{eff} is indeed a robust metric for generalisation, comparing favourably to other norm- and flatness-based measures.

While the spirit of these works is correct in seeking an alternative to parameter counting, the mathematical basis for using the Hessian only makes sense for regular models and therefore not for neural networks. Recent

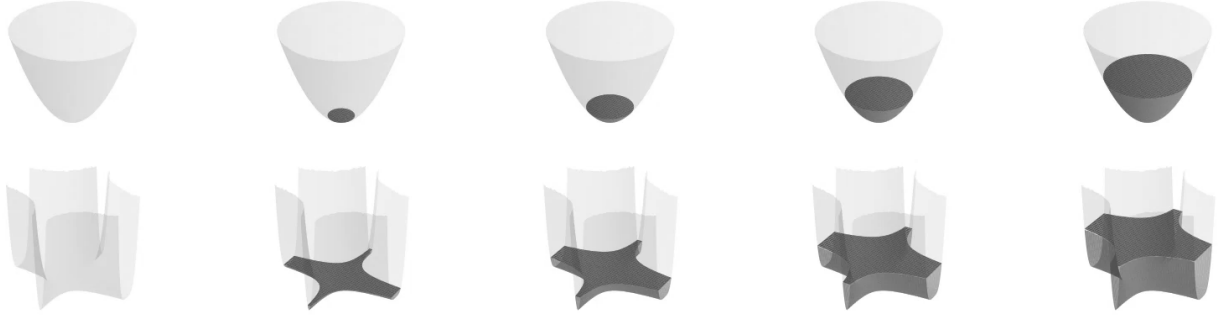


Figure 52.1: The LLC $\lambda(w^*)$ measures volume scaling around a local minimum of the loss. Top: for the regular quadratic loss $K(w) = w_1^2 + w_2^2$, the area (or “2D-volume”) of the near-optimal set $\{K(w) < \varepsilon\}$ around $w^* = (0, 0)$ scales as ε , so $\lambda(w^*) = 1$ (equal to $d/2$ with $d = 2$). Bottom: for the singular loss $K(w) = w_1^2 w_2^4$, which vanishes on the entire union of the two coordinate axes, the near-optimal set is a cross-shaped neighbourhood of the axes, and its area scales as $\varepsilon^{1/4}$, so $\lambda(w^*) = 1/4$. A smaller LLC means more volume near the minimum: as $\varepsilon \rightarrow 0$, far more parameter values achieve loss below ε than in the regular case. (Adapted from Figure 2 in Lau et al. [978]; images reproduced with permission from Hoogland and van Wingerden [782].)

advances in SLT provide a more rigorous definition of effective dimensionality, given by the local learning coefficient [978], which we will introduce below.

From global to local. The learning coefficient defined above is a *global* property of the *bottom of the loss landscape*: it describes the overall scaling of volume near all global minima of $K(w)$ combined, and the smallest λ among them dominates the asymptotics. However, in complex models such as neural networks, the geometry of $K(w)$ may vary widely across different minima, symmetry classes, or training trajectories. Some regions of parameter space may be flatter or more degenerate than others. Moreover, we cannot usually assume that training reaches a global minimum. To capture this variation, Lau et al. [978] define and develop a *local* analogue of the learning coefficient.

52.4.1 Local learning coefficient

The *local learning coefficient (LLC)* (or local RLCT) $\lambda(w^*)$ is defined in a completely analogous way to the global learning coefficient except we focus on a local minimum w^* of $L(w)$ (or equivalently of $K(w)$). Let $B(w^*)$ be a small closed ball centred on w^* such that for all $w \in B(w^*)$ we have $L(w) \geq L(w^*)$. Instead of the set of almost optimal parameters, we consider the set $\{w \in B(w^*) : L(w) - L(w^*) < \varepsilon\}$. Then the volume definition of the global learning coefficient can be directly extended, as the constant $\lambda(w^*)$ controlling the asymptotics of the prior-weighted volume:

$$\int_{w \in B(w^*) : L(w) - L(w^*) < \varepsilon} \varphi(w) dw \sim C \varepsilon^{\lambda(w^*)} (-\log(\varepsilon))^{m(w^*)-1} \quad \text{as } \varepsilon \rightarrow 0. \quad (52.13)$$

Figure 52.1 illustrates this volume scaling for a regular (quadratic) loss and for a singular one.

LLC estimator. One can adapt the WBIC estimator of the LML discussed above to estimate the LLC. For details, the reader should consult Lau et al. [978]. First let us define the local-tempered posterior

$$p(w|w^*, \beta, \gamma) \propto \exp \left\{ -n\beta L_n(w) - \frac{\gamma}{2} \|w - w^*\|_2^2 \right\}, \quad (52.14)$$

where $\beta > 0$ is an inverse temperature and $\gamma > 0$ indicates how close we want to stay to w^* . Let us denote the expectation of a function $f(w)$ with respect to the posterior distribution in (52.14) as

$$\mathbb{E}_{w|w^*, \beta, \gamma}[f(w)] := \int f(w) p(w|w^*, \beta, \gamma) dw.$$

The **local learning coefficient** estimator is given by

$$\hat{\lambda}(w^*) := n\beta^* [\mathbb{E}_{w|w^*, \beta^*, \gamma}[L_n(w)] - L_n(w^*)]. \quad (52.15)$$

where $\beta^* = 1/(\log n)$. The fact that this estimates $\lambda(w^*)$ follows from Equation (52.11).

Comparing N_{eff} and the LLC. A minimal example separates the two quantities. Consider the loss $K(w) = w_1^2 w_2^4$ from Figure 52.1. At the minimum $w^* = (0, 0)$ every second derivative of K vanishes, so the Hessian of K at w^* is the zero matrix and N_{eff} evaluated on it equals 0 for every $z > 0$. The eigenspectrum reports that no direction is constrained, and it reports the same value for $w_1^4 w_2^6$ or for any other loss whose quadratic part vanishes at the minimum, so it cannot tell these geometries apart. The LLC does distinguish them: the monomial formula of Section 52.2.2 gives $\lambda(w^*) = 1/4$ for the first loss and $\lambda(w^*) = 1/6$ for the second. The reason is that the Hessian captures only the quadratic part of the loss, which is precisely the part that vanishes at a singularity, while the LLC measures the volume scaling of the full function. One caveat in interpreting this comparison: in practice N_{eff} is computed from the Hessian of the empirical loss L_n at a trained network $\hat{w}$, which will generally have small nonzero eigenvalues rather than vanish exactly. The comparison above concerns the population loss exactly at the singularity, the same idealisation used throughout this chapter. The point survives the idealisation: near a singular minimum, the eigenvalues of the Hessian convey how nearly degenerate the quadratic form is, while the LLC quantifies the degeneracy itself.

Applications of the LLC. The LLC estimator has been developed into a practical diagnostic for studying the internal geometry and developmental dynamics of deep learning models. Hoogland et al. [783] apply the LLC to transformer training and show that shifts in local complexity coincide with discrete phases of representational development, linking changes in loss-landscape degeneracy to the formation of new computational subsystems. Building on this framework, Wang et al. [1705] introduce the *refined* LLC to analyse attention heads individually, demonstrating that differentiation and specialisation among heads emerge through measurable reductions in local complexity. Together, these studies use the LLC to reveal how structure and function in deep networks unfold through geometric transitions in the learning process, bridging singular learning theory with observable deep training phenomena.

Takeaways:

Together, the (global) learning coefficient λ and its local refinement $\lambda(w^*)$ describe how model complexity and posterior concentration emerge from the geometry of the parameter–likelihood map. They give SLT its distinctive power: complexity and generalisation are determined not by parameter count or curvature, but by how the *volume* of near-equivalent models collapses under the data.

52.5 Misconception 3: The benefits of marginalisation are not about predictive performance

The motivation for using Bayesian marginalisation in deep learning has traditionally been twofold: for philosophical coherence with Bayesian principles, and more practically, for its ability to produce reliable **uncertainty quantification**. The resulting posterior predictive distribution naturally encodes uncertainty, distinguishing it from point-estimate methods such as MAP or MLE. However, the conversation around raw **predictive performance** (i.e., generalisation error) is more muddled. It is often implicitly assumed that any accuracy gain from the computationally expensive process of marginalisation is modest. In simple, “regular” models, this is largely true, leading to the belief that one pursues marginalisation for the uncertainty estimates, not for the predictions themselves.

This section aims to show that this picture is incomplete. We will demonstrate that for the singular models that dominate modern machine learning, marginalisation may provide significant advantage in predictive performance itself. The basis for this advantage can be understood using SLT, which offers a rigorous foundation for this powerful, yet underappreciated, aspect of marginalisation, or Bayesian averaging. The results in this section concern the exact Bayes PPD of Equation (52.1), in which the intractable posterior is integrated exactly; whether practical approximations to the posterior inherit the advantage is a separate question, to which we return at the end of this section and in Misconception 4 (Section 52.6).

To formalise this, we must first define the three predictive distributions in question, all derived from a training dataset $\mathcal{D}_n = \{x_i\}_{i=1}^n$:

- **Bayesian PPD:** In Bayesian learning, we are uncertain about which set of model parameters is the correct one. Instead of committing to a single “best” set, marginalisation is the process of **averaging the predictions of all possible models**. Each model’s prediction is weighted by its posterior probability—how

plausible that set of parameters is, given the training data. This is captured by the Bayesian PPD in (52.1). Here, we are “integrating out” or “marginalising out” the parameters w .

- **MAP Predictive:** This is a point estimate based on the maximum of the posterior.

$$\hat{p}_{\text{MAP}}(x) := p(x \mid \hat{w}_{\text{MAP}}), \quad \hat{w}_{\text{MAP}} := \arg \max_{w \in \mathcal{W}} \pi(w \mid \mathcal{D}_n).$$

- **MLE Predictive:** This is a point estimate based on the maximum of the likelihood.

$$\hat{p}_{\text{MLE}}(x) := p(x \mid \hat{w}_{\text{MLE}}), \quad \hat{w}_{\text{MLE}} := \arg \max_{w \in \mathcal{W}} \prod_{i=1}^n p(x_i \mid w).$$

We shall measure the performance of any such predictive distribution $\hat{p}(x)$ using the following notion of generalisation error:

$$G(\hat{p}) := \text{KL}(p_0(x) \parallel \hat{p}(x)).$$

That is, G is the KL divergence from the true data-generating distribution p_0 to $\hat{p}$. This is what is known, up to an additive constant, as “negative log likelihood” or “test log likelihood” in the Bayesian deep learning literature. We also note that this is not the typical generalisation error used in ML benchmarking tasks such as percentage correct on the test set.

We denote the generalisation errors for our three methods as $G_{\text{Bayes}}(n) := G(\hat{p}_{\text{Bayes}})$, and similarly $G_{\text{MAP}}(n)$ and $G_{\text{MLE}}(n)$, making the dependence on the sample size explicit. SLT provides the asymptotic forms for these errors. For the Bayes PPD, we have [1726, Thm. 15]

$$\mathbb{E}_{\mathcal{D}_n} G_{\text{Bayes}}(n) = \frac{\lambda}{n} + o\left(\frac{1}{n}\right) \quad (52.16)$$

where λ is the global learning coefficient and $\mathbb{E}_{\mathcal{D}_n}$ indicates expectation over the training data drawn from the true p_0 . The fact that the same learning coefficient controls both the asymptotics of the LML $\log Z_n$, as in Equation (52.10), and of $G_{\text{Bayes}}(n)$, i.e. both model complexity and generalisation error, is an important feature of SLT¹⁰. While not our main focus, we briefly note that the generalisation error can be estimated using only training data with the Widely Applicable Information Criterion (WAIC) [1724]. PAC-Bayesian analysis provides a complementary frequentist route to generalisation guarantees for Bayesian and randomised predictors, with bounds that hold at finite sample size; see Chapter 22.

For point estimates, the situation is different. We have [1723, Thm. 6.4]

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_n} G_{\text{MAP}}(n) &= \frac{C_{\text{MAP}}}{2n} + o\left(\frac{1}{n}\right) \\ \mathbb{E}_{\mathcal{D}_n} G_{\text{MLE}}(n) &= \frac{C_{\text{MLE}}}{2n} + o\left(\frac{1}{n}\right) \end{aligned}$$

where the constants $C_{\text{MAP}}, C_{\text{MLE}} > 0$ are distinct but both related to expectations of the maximum value of the square of the Gaussian process $\xi(w)$ on W_0 . If the model is regular, then $C_{\text{MAP}} = C_{\text{MLE}} = d/2$, recovering the classical asymptotics. In singular models, the maximum of ξ^2 is taken over the positive-dimensional set W_0 , and maxima of Gaussian processes grow with the size of the index set; this mechanism pushes both constants above $d/2$, and hence above λ . We caution that this comparison is an observation discussed by Watanabe [1723] rather than a general theorem: exact values of C_{MAP} and C_{MLE} are known only in special cases.

This means two things. First, we can expect the Bayes PPD to have generalisation error different from that of MAP and MLE. Second, we can expect in general for the Bayes PPD to achieve a smaller leading constant in its $1/n$ generalisation error. This advantage arises because Bayesian averaging explores the entire set of optimal parameters (W_0), smoothing out any overfitting to one specific solution, while MAP/MLE commits to a single point estimate found on the noisy training data.

Example: finite mixture of Gaussians. Consider a mixture of r isotropic unit variance Gaussian components in $\mathbb{R}^p$, fit to data from a mixture with $r_0 < r$ components. This model is singular because there are many ways to configure the redundant components to produce the same optimal solution (e.g., by swapping component labels

¹⁰This relation is itself less mathematically deep than either of those asymptotics.

or setting weights to zero). In this case, the learning coefficient λ is strictly smaller than $\frac{d}{2} = \frac{1}{2}(pr + r - 1)$. In fact, one can show in this case [1723, Section 7.3, equation (7.3)] that

$$\lambda \leq \frac{1}{2}(pr_0 + r - 1). \quad (52.17)$$

As a result, $G_{\text{Bayes}}(n) \approx \lambda/n$ is smaller than the MAP/MLE generalisation error, which does not fully exploit the symmetry in W_0 .

The same principle applies directly to deep learning. The symmetries and redundancies in neural networks (e.g., permutation of neurons) create vast, flat sets W_0 of equivalent parameters. Bayesian prediction effectively averages over these equivalent solutions, reducing generalisation error in proportion to the model's true complexity, λ . Point estimates cannot exploit this geometry and therefore suffer a higher generalisation error.

Despite these theoretical advantages, performing Bayesian marginalisation faithfully in modern deep networks remains an open challenge. Several studies have shown that popular approximations often fall short of true posterior averaging. For instance, variational methods fail to capture posterior correlations [542], and empirical analyses reveal that approximate posteriors such as SWA-Gaussian or Laplace differ substantially from true ones [826]. Other works have critiqued Bayesian interpretations of dropout (Chapter 14) [536] and deep ensembles [1756] and shown that many practical improvements stem from implicit ensembling rather than accurate marginalisation [1116]. Misconception 4 (Section 52.6) makes this gap precise in the setting of variational inference: the quality of a posterior approximation as measured by the ELBO and the quality of its predictive distribution are governed by different asymptotic constants.

52.6 Misconception 4: Good ELBO means good predictive performance

Variational inference (VI; see Chapter 12 for an introduction) remains one of the most widely used and theoretically appealing approaches to approximate Bayesian deep learning. It offers a principled optimisation objective — the evidence lower bound (ELBO) — and, unlike more ad hoc methods such as deep ensembles (Chapter 19), VI is directly connected to Bayesian principles via a Kullback–Leibler projection onto a tractable family of approximate posteriors. This combination of conceptual clarity and scalability has made VI a popular alternative to MCMC.

In Bayesian deep learning, VI is often used to approximate the PPD. It might be natural to expect that improving the ELBO will also improve the resulting predictive distribution: since the ELBO lower-bounds the log marginal likelihood, tightening the bound should bring the approximation closer to the true posterior, and hence to the true Bayesian predictive distribution. In *regular* models, where the posterior is asymptotically Gaussian, this expectation can often be justified. However, in *singular* models — which include all modern deep neural networks — the relationship between ELBO and predictive performance can break down. One variational approximation may achieve a *smaller* ELBO gap than another, yet still produce *worse* predictions. This exposition is based mainly on Wei and Lau [1732] and we largely follow the notation and terminology therein.

Setup. We work in the supervised setting, unlike in previous sections of this chapter. To approximate the intractable posterior distribution, VI requires two ingredients: a variational family and a divergence. Let $\mathcal{Q}$ denote a variational family. We present standard VI, which uses the KL divergence, leading to the optimisation problem

$$q^* \in \arg \min_{q \in \mathcal{Q}} \text{KL}(q(w) \parallel \pi(w \mid \mathcal{D}_n)).$$

Note this is equivalent to *minimising* the (normalised) variational free energy (VFE),

$$\bar{F}_{\text{vb}}(n) = \mathbb{E}_q[nK_n(w)] + \text{KL}(q(w) \parallel \varphi(w)),$$

where φ is the prior and $K_n(w)$ is the empirical KL divergence. The **VFE is equal to the ELBO**, up to an additive constant independent of q and a sign. So while we maximise the ELBO, we seek to minimise the VFE. Let us denote the optimum value of the VFE

$$\bar{F}_{\text{vb}}^*(n) = \min_{q \in \mathcal{Q}} \bar{F}_{\text{vb}}(n).$$

Once we have obtained our variational approximation q^* to the posterior distribution, we can use it for prediction. Let us call this the variational predictive distribution, which is given by

$$\hat{p}_{\text{vb}}(y \mid x) := \int p(y \mid x, w) q^*(w) dw.$$

Let us denote the variational generalisation error (VGE),

$$G_{\text{vb}}(n) := \mathbb{E}_x \text{KL}(p_0(y | x) \| \hat{p}_{\text{vb}}(y | x)),$$

where p_0 is the true conditional distribution of y given x and the expectation is over the input distribution.

Two different asymptotic coefficients. In singular models, the minimum VFE has the following asymptotic expansion

$$\mathbb{E}_{\mathcal{D}_n} \bar{F}_{\text{vb}}^*(n) = \lambda_{\text{vfe}} \log n + o(\log n),$$

while the VGE admits

$$\mathbb{E}_{\mathcal{D}_n} G_{\text{vb}}(n) = \frac{\lambda_{\text{vge}}}{n} + o\left(\frac{1}{n}\right).$$

The constants λ_{vfe} and λ_{vge} describe *different* aspects of the approximation:

- λ_{vfe} governs the $\log n$ term in the minimum VFE and hence the optimal ELBO.
- λ_{vge} governs the $1/n$ term in the generalisation error of the downstream predictive distribution that uses the variational approximation to the posterior.

There is, unfortunately, no reason for λ_{vfe} and λ_{vge} to coincide in a singular model, unlike for standard Bayesian inference. A variational family can have a *smaller* VFE gap (i.e. λ_{vfe} closer to the optimal learning coefficient λ) but a *larger* predictive error (larger λ_{vge}).

The clearest concrete setting is reduced rank regression, the linear network model of Section 52.2.2 with one hidden layer. For Gaussian mean-field variational families, Nakajima and Watanabe [1221] derived the asymptotics of both the minimum VFE and the VGE in closed form, and the two coefficients genuinely differ: depending on the input and output dimensions, the number of hidden units, and the rank of the true map, either $\lambda_{\text{vfe}} > \lambda_{\text{vge}}$, in which case minimising the VFE offers control over the VGE, or $\lambda_{\text{vfe}} < \lambda_{\text{vge}}$, in which case even a small variational approximation gap is compatible with an induced predictive distribution with high generalisation error. The same inversion arises when comparing two variational families: Wei and Lau [1732] illustrate schematically (their Figure 2(b)) that the family with the smaller VFE gap can induce the predictive distribution with the larger λ_{vge} , and some of their experiments with tanh networks exhibit exactly this scenario, with the ordering of two families by ELBO opposite to their ordering by predictive performance.

Takeaways:

In singular Bayesian deep learning, it is *natural* to expect that a better ELBO will lead to better predictions, but this is not guaranteed. ELBO and predictive accuracy are governed by different asymptotic constants, and can be ranked in opposite directions.

52.7 Summary

Singular learning theory (SLT) provides a coherent asymptotic framework for Bayesian deep learning, clarifying where classical intuitions break down. It explains why local quadratic (Laplace) approximations fail in neural networks, why model complexity cannot be read off from parameter counts or Hessian spectra, why Bayesian marginalisation can improve predictive accuracy rather than merely uncertainty estimates, and why optimising the ELBO does not necessarily yield a good predictive distribution. Across these settings, SLT replaces classical asymptotics based on curvature with a geometry-based description of how the posterior and marginal likelihood behave.

Chapter 53

Active Learning and Experimental Design

Today we have powerful methods for using data once we have it [450, 1212, 1213, 1632]. Yet the way we *gather* data in the first place remains strikingly low-tech, from the design of clinical trials [1075, 1413, 1431] to the construction of model evaluations [249, 1042, 1550]. This is a significant bottleneck. We are limited by the number of human experts able to effectively direct data-gathering, but also by the capacity of even the most capable experts to handle masses of information and complex tradeoffs. We are thus underequipped to gather an essential resource: good data is central to scientific progress [389, 555, 556, 696, 697], is a core requirement for successful machine learning [438, 701, 1037], and more broadly influences the quality of decision-making across society [505, 1161, 1670].

The challenge of addressing the data-gathering bottleneck—and the payoff of enhancing a whole spectrum of data-driven activities, multiplying the value they produce—motivates research on *experimental design*. This formalises the data-gathering problem: the design of an abstractly defined experiment represents aspects of data generation that we can control, such as which questions to ask a user or how to configure lab equipment; the choice of design influences the data produced by the experiment, such as user responses or lab measurements; and our aim is to choose the best design we can, where rigorously defining “best” is a central challenge. More generally we can think of designing a whole sequence of experiments, with our task then being to optimise a design policy that specifies how to design each experiment based on the data we have already gathered.

The study of experimental design has evolved considerably since early formal work by Gergonne [608] and Peirce [1310]. Spurred on by Fisher [525, 526] and Smith [1525], it grew significantly within the field of statistics in the twentieth century [165, 198, 290, 291, 395, 396, 510, 886, 1049, 1050, 1403, 1692].¹ Recent years have seen yet more growth as practical methods have improved [790, 1369, 1443]. Alongside work within statistics, there has been interest and innovation within machine learning, especially since the late 1980s, often under the banner of *active learning* [60, 321, 322, 1106, 1107] but also associated with topics including curiosity-driven learning [1470, 1481], exploration [1202, 1604, 1673] and query-based learning [34, 104, 1652].

This research has yielded concrete progress. In machine learning we have seen promising demonstrations of automated data-gathering for image classification [1237], model evaluation [1464], reward modelling [694], robot control [106] and text classification [1019]. Meanwhile in the sciences we have seen successful investigations in biology [866], chemistry [671], neuroscience [1497], physics [475] and psychology [1651].

Further work is nevertheless needed. While existing proof-of-concept studies are encouraging, there remains a substantial gap between them and widespread deployment of automated data-gathering. Real-world problems are more complex and diverse than the problems that have been studied so far. Solving them will require methods that can be flexibly tailored to different problems and that can be relied upon to perform well.

In this chapter we aim to lay the groundwork for future advances. We begin by covering some foundational ideas from statistical decision theory, particularly how the contrasting Bayesian and frequentist attitudes towards observed data lead to different notions of optimal decision-making. Following this we translate optimality criteria for general decisions into optimality criteria for experimental design, and we discuss some important challenges in bridging from theory to practice, including instantiating concrete realisations of abstract constructs and approximating the solutions to intractable optimisation problems. We then present active learning as a particular form of experimental design focused on accurate predictions (more specifically predictions of outputs for inputs that are often unknown at the time of data-gathering), present some of the most popular methods in the field, and discuss some key methodological considerations in active learning.

¹See Garnett [584] for an excellent history.

53.1 Decision-theoretic foundations

Before directly addressing data-gathering, we first lay out the theoretical foundations of more general decision-making. The rationale for this is that many of the core principles of effective data-gathering require a deeper understanding of decision theory than just the well-known notions of Bayesian inference and empirical-risk minimisation.

At a high level, statistical decision theory concerns how to evaluate and optimise policies, $\pi : \mathcal{D} \rightarrow \mathcal{A}$, that map from data, $d \in \mathcal{D}$, to actions, $a \in \mathcal{A}$ [123, 166, 395, 514, 1050, 1364, 1405, 1463, 1694]. The *consequences* of these actions will depend on some unknown state of the world, $z \in \mathcal{Z}$, and our aim is generally to minimise the loss (or equivalently maximise the utility) of these consequences. Notably this formalism encompasses both concrete real-world actions, such as prescribing a treatment to a patient, and more abstract actions, such as estimating a parameter in a model.

Frameworks to solve such problems can generally be broken into two branches, *Bayesian* and *frequentist*, each of which we will cover. Both use notions of expected loss to evaluate policies, but they differ in their treatment of observed data, d , and the unknown world state, z . While inevitably linked to the respective Bayesian and frequentist definitions of probability (i.e. subjective beliefs under incomplete information and long-run relative frequency of an event), it is important to appreciate that these decision-theoretic formalisms extend far beyond this more philosophical distinction and have profound implications across modern statistics and machine learning.

53.1.1 Bayesian decision theory

The underlying question of Bayesian decision theory is *how should an agent choose actions in the presence of incomplete information?* It considers this problem from the point of view of the agent itself, who must formulate probabilistic beliefs about the world and then make decisions about how to best act in light of these.

Mathematically, a decision-making problem under the Bayesian framework is defined by two terms: a belief state about the world given the observed data expressed in the form of a probability distribution over states, $p(z|d)$, and a loss function (or equivalently negative utility function), $\ell : \mathcal{A} \times \mathcal{Z} \rightarrow \mathbb{R}$, that measures the cost of taking action a if the true state of the world is z . Given these, we can evaluate actions using their Bayesian expected loss, defined as follows.

Definition 53.1 (Bayesian expected loss). *The Bayesian expected loss, $L_B(a, d)$, of taking action $a \in \mathcal{A}$ given observed data d , is the expectation over an agent's subjective belief distribution on world states, $z \sim p(z|d)$, of an integrable and measurable loss function, $\ell : \mathcal{A} \times \mathcal{Z} \rightarrow \mathbb{R}$, that represents its preferences over action-state pairs:*

$$L_B(a, d) = \mathbb{E}_{p(z|d)}[\ell(a, z)]. \quad (53.1)$$

The underlying justification for this Bayesian-expected-loss formulation will be discussed in 53.1.1. For now we just note the importance of the loss function needing to represent the agent's *preferences*: the choice of the loss function is not arbitrary for a given agent. In fact, in a canonical derivation of the Bayesian-expected-loss formulation [1463], both the beliefs and loss function are simultaneously derived from an underlying system of rational preferences.

Here the data, d , is an observed quantity and is treated as fixed in the Bayesian framework. As such, the Bayesian expected loss is effectively just a function of our action and so can be used to define the notion of a Bayes-optimal action (also known as a Bayes-optimal decision, Bayes action, or, in the context of estimation, a Bayes estimator) as follows.

Definition 53.2 (Bayes-optimal action). *An action, $a_B \in \mathcal{A}$, is known as Bayes-optimal for a given belief distribution, $p(z|d)$, and loss function, $\ell : \mathcal{A} \times \mathcal{Z} \rightarrow \mathbb{R}$, if it optimises the Bayesian expected loss, that is²*

$$a_B = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{p(z|d)}[\ell(a, z)]. \quad (53.2)$$

We can further extend this idea of a Bayes-optimal action to that of a Bayes-optimal *policy*, $\pi_B : \mathcal{D} \rightarrow \mathcal{A}$, that maps from any given realisation of the data, d , to a corresponding Bayes-optimal action.

Definition 53.3 (Bayes-optimal policy). *A policy, $\pi_B : \mathcal{D} \rightarrow \mathcal{A}$, is known as a Bayes-optimal policy if it produces Bayes-optimal actions for any given input data, d :*

$$\pi_B(d) = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{p(z|d)}[\ell(a, z)] \quad \forall d \in \mathcal{D}. \quad (53.3)$$

²In the interest of clarity, we implicitly assume here and throughout that there is a unique minimiser of the Bayesian expected loss for any given d . In practice, the minimiser might not be unique, so formally we should say that any $a_B \in \arg \min_{a \in \mathcal{A}} \mathbb{E}_{p(z|d)}[\ell(a, z)]$ is a Bayes-optimal action.

One of the key upshots of 53.2 is that the data only influences the Bayes-optimal action through the belief over states, $p(z|d)$. We can thus think of a Bayes-optimal policy as mapping from beliefs to actions, rather than requiring it to be a mapping from the data itself. This is arguably why the Bayesian-statistics and Bayesian-machine-learning literatures are so overwhelmingly focused on estimating posteriors [23, 170, 826, 874, 1609, 1742]: our beliefs on z contain everything we need to know to make downstream decisions (unlike in frequentist frameworks, where we need to reason about other data we *could* have seen). It is also crucial for sequential decision-making, where it shows that we need only keep track of our current beliefs to make optimal sequential decisions.

It is important, though, to appreciate that we have not made any direct assumptions on how the belief state, $p(z|d)$, is derived in formulating this decision-theoretic framework. As such, while Bayes' rule separately gives a mechanism for how beliefs should be updated to ensure they are coherent, this framework for making *static decisions* still holds up when beliefs have been derived by some other mechanism, as long as they really do represent the beliefs of the agent. Thus, we do not need to have explicitly defined prior and likelihood to follow a Bayesian approach to decision-making, at least for static decisions. Instead, we might, for example, directly define our belief state after seeing data, take it from some foundation model that we input the data to, or even take the belief state as the output from some training procedure on the data. As such, for static settings we should arguably use the notation $p(z; d)$ instead of $p(z|d)$, to emphasise that the data is taken as fixed and we want to construct our most faithful beliefs using it, but do not necessarily need to explicitly construct a joint model on z and d (whereas in the experimental-design setting we will find that such a joint *is* explicitly required).

Justification and interpretation

At a high level, the two core principles that lead to the above formulation of the Bayesian expected loss (and by extension Bayes-optimal actions) are *coherence* [383, 1376] and *rational preference* [1687].

Coherence is the idea that *subjective beliefs* follow the standard axioms of probability [912], namely non-negativity, additivity and normalisation. This statement is equivalent to saying that beliefs do not permit the construction of a Dutch book [383], wherein the agent can make a set of bets according to their beliefs that would guarantee they lose money. Cox's theorem [338] provides a further direct justification of coherence by showing that any *logical* approach for reasoning about beliefs must follow the laws of probability, where this notion of logical reasoning is defined through a set of intuitive axioms. This in turn implies that conditioning on observations yields Bayesian updates in beliefs: Bayes' rule is formally a result about probabilities and not beliefs, but by showing that logical beliefs must behave like probabilities, Cox's theorem allows us to apply Bayes' rule to belief updating.

Rational preference is the idea that the agent's preferences between different "lotteries" themselves adhere to natural logical rules, where each lottery is a given distribution over possible outcomes (e.g. an agent might prefer a lottery that assigns probabilities 0.3 and 0.7 respectively to the events that it rains and does not rain tomorrow, to a lottery that assigns probabilities 0.6 and 0.4). Specifically, von Neumann and Morgenstern [1687] showed that if preferences follow the axioms of completeness, transitivity, continuity and independence, then they can be represented through some utility function, such that the agent prefers lottery A over lottery B if and only if the expected utility of A exceeds that of B . In transferring these ideas to the Bayesian decision formulation, we take the space of outcomes to be the *consequences* that result from a given action-state pair, (a, z) , and the loss is then simply the negative of the utility function implied by our preferences between such consequences. Note, though, that in this VNM utility theory the expectation is over an "objective" stochasticity in outcomes, with the key conceptual step needed to go from this to Bayesian decision theory being to show the axioms still imply an expected-utility formulation when the expectation is instead over subjective beliefs.

Together these axioms are already enough to derive the form of the Bayesian expected loss if one is also willing to accept the idea of subjective probabilities existing independently of preferences: coherence ensures that subjective beliefs behave like real probabilities and we can then use VNM utility theory to show that our preferences can in turn be represented as an expected loss. Savage [1463] was able to take things a step further and show how we can directly derive *both* subjective beliefs and a utility function together from a rational agent's preferences under uncertainty, subject to additional, slightly more complex axioms on nontriviality, measurability, richness, consistency and separability. In essence, Savage's work shows that if rational preference is defined in this stronger sense then coherence does not need to be a separate assumption: it is a property that directly follows from his axioms. The idea that we should update our beliefs using Bayes' rule in a sequential setting follows in turn from this need to be coherent to be rational.

The idea of deriving beliefs from preferences may at first seem a little odd. However, it becomes much more natural when we consider the idea of having preferences between different *bets*. For example, if we prefer a bet that returns £10 if it rains tomorrow to a bet that returns £10 if it does not, this implies that we *believe* it is more likely to rain tomorrow than not. Extending this idea to preferences over all possible bets will then yield a

full belief distribution, with rational preferences leading to coherent beliefs to avoid Dutch books. For example, if we are completely scale-insensitive in our preferences (i.e. our preferences only depend on the ratio of returns, not the stake) and prefer a bet returning $\mathcal{L}X$ if it rains tomorrow over a bet returning $\mathcal{L}10$ if it does not if and only if $X \geq 5$, then this implies we believe the probability of it raining tomorrow is exactly $2/3$.

The key intuition to take away from Savage’s results is that any “rational” agent can implicitly be represented as a Bayes-optimal policy that optimises some Bayesian expected loss, where the quotations on “rational” are used to indicate this is meant in the formal sense of the agent’s preferences following the Savage axioms. Note that this is not the same thing as saying that we can only derive rational agents from Bayesian assumptions or that an agent following any given Bayesian formulation will behave sensibly for a particular problem. It simply means that if our agent takes actions in a “rational” manner then it must have some implicit Bayesian-expected-loss representation, i.e. there must be some belief–loss pairing under which it is the Bayes-optimal policy.

Criticisms

There is a wide variety of criticisms of the Bayesian decision framework and it is only feasible to provide a very brief summary of them here.³ We emphasise that many of these criticisms are disputed by people who advocate Bayesian approaches, with the Bayes-vs-frequentist debate representing by far the largest historical disagreement in the statistics community, at times bordering on a full-blown schism.

There are firstly various largely uncontroversial practical criticisms of Bayesian approaches regarding difficulties in effectively deriving belief states, the intractability of exact inference in most problems, and other computational issues. In particular, the need to derive a full belief distribution over z can be a lot more challenging than just deriving a point estimate. The scale of these challenges will naturally vary and they need to be considered on a problem-by-problem basis.

Most of the more fundamental criticisms of Bayesian decision theory stem from objections to the use of subjective beliefs to guide decision-making, or as it is more commonly put, objections to placing subjective priors on the state of the world [485]. These objections can themselves vary from more philosophical objections [1017] about subjective decision-making itself (e.g. that we should instead strive to make decisions with “objective” frequentist guarantees) to more pragmatic objections about our ability to construct probabilistic models that represent actual internal beliefs and account for all possible eventualities [858]. With regard to the latter, it is clear that just because an “optimal” agent must have a Bayesian representation does not mean that using a particular Bayesian *model* will yield effective behaviour in practice; most modern scholars would agree that there are some problems where a Bayesian approach is not practically sensible and others where it can be effective.

The former fundamental objections, though, are arguably more controversial. The Bayesian decision framework is a *normative* one specifying *idealised* decision-making, rather than a descriptive framework with respect to real human behaviour or a prescriptive framework for constructing effective decision-making systems in practice. One can, of course, dispute the practical utility of such a normative framework; but to dispute its normative statement that rational agents are Bayes-optimal requires criticising the Savage axioms themselves, as the statement directly follows from these.

Regardless, it is important to appreciate that using many modern Bayesian machine-learning methods—including virtually all Bayesian neural networks (BNNs)—to make decisions and predictions actually *violates* the assumptions of the Bayesian decision framework. Bayesian decision-making is fundamentally rooted in *beliefs* and when we, say, place a prior over the huge number of parameters of a neural network, it is not reasonable to assume this actually corresponds to our internal beliefs. By extension, any posterior we derive from this prior is also unlikely to be a true representation of our beliefs and the decisions we make from it unlikely to be Bayes-optimal with respect to our true beliefs. This is not to say that taking a Bayesian approach in such cases is foolish, or even that there are necessarily better ways of deriving our beliefs, but simply that we need to be careful in how we interpret the “posteriors” and decisions arising from such models. In particular, we should not blindly assume in such settings that the posterior is the best thing to take an expectation with respect to when it comes to making decisions. Moreover, the inexact nature of approximate inference can further push the behaviour of such models away from the Bayesian ideal; for example, it has recently been shown that common BNN approaches with approximate-inference schemes exhibit problematic behaviours that true Bayesian methods should not have [1329].

53.1.2 Frequentist decision theory

The Bayesian approach to making decisions that we have just laid out was rooted in how a rational agent should make decisions in the presence of incomplete knowledge. The frequentist view, by contrast, tries to remove

³See e.g. Berger [123], Efron [485], Gelman [597] and Rainforth [1365] for more comprehensive discussions.

subjectivity from the problem and instead defines the optimality of decision rules in terms of expected or long-run performance over *possible data that might have been generated*. At a high level, one can usually think of Bayesian methods as probabilistically reasoning about the state of the world, $z \in \mathcal{Z}$, while frequentist methods take this to be fixed and probabilistically reason about what data, $d \in \mathcal{D}$, we *could* have observed for a given z .

In other words, the frequentist approach takes an unconditional perspective towards observed data: d is treated as one of many possible realisations, and *decision rules* (i.e. policies), $\pi : \mathcal{D} \rightarrow \mathcal{A}$, are evaluated in terms of their expected performance in expectation over possible d , using the *frequentist risk* defined as follows.

Definition 53.4 (Frequentist risk). *The frequentist risk, $R(\pi, z)$, of a decision rule, $\pi : \mathcal{D} \rightarrow \mathcal{A}$, given a world state, $z \in \mathcal{Z}$, and a data-generating distribution, $p(d; z)$, is the expectation over data, $d \sim p(d; z)$ (and any stochasticity in the decision rule itself) of a loss function, $\ell : \mathcal{A} \times \mathcal{Z} \rightarrow \mathbb{R}$:*

$$R(\pi, z) = \mathbb{E}_{p(d; z)}[\ell(\pi(d), z)]. \quad (53.4)$$

Here the concept of a loss function is shared with that in Bayesian decision theory, again with underlying motivations from VNM utility theory. However, $p(d; z)$ is not interpreted as a belief distribution but instead as a true data-generating distribution, with approximations to this then required for practical estimation of the frequentist risk (e.g. assuming a particular model form, typically known as a likelihood). Note our deliberate use of the notation $p(d; z)$, as opposed to $p(d|z)$, to represent the fact that z is not being treated probabilistically so we cannot think about probabilistically conditioning on it.

Unlike the Bayesian setup—where optimal actions were fully specified by the model and loss—the frequentist risk is a function of both π and z . Choosing a policy in the frequentist setup therefore still requires us to either specify some particular value of z or reason over the possible values that z could take. Because of this, there is no singular notion of a “frequentist-optimal policy” in the same way as there is a Bayes-optimal policy and we instead need to explicitly define an *optimality criterion* on top of the definition of the frequentist risk in order to choose a policy. There is significant flexibility in how this optimality criterion is set up in the frequentist framework and we essentially have free rein in what we choose for it, provided it only ever returns *admissible* decision rules as defined below.

Definition 53.5 (Admissible decision rule). *A decision rule, or policy, π , is admissible for a given frequentist-risk formulation if it is not dominated by any other policy across all possible z , that is*

$$\nexists \pi' : (\forall z : R(\pi', z) \leq R(\pi, z)) \wedge (\exists z : R(\pi', z) < R(\pi, z)). \quad (53.5)$$

Below we highlight some common optimality criteria used in practice.

- **Local optimality:** assuming a particular value of z (e.g. a maximum-likelihood estimate) and then choosing a policy that is optimal for this chosen value.
- **Uniform optimality:** under certain model and loss setups (or assumptions) we can sometimes find policies that are optimal for all z (e.g. confidence intervals under the assumption estimates are Gaussian distributed about the true value).
- **Minimax optimality:** choosing a policy that has the best worst-case performance over some allowed set of z , i.e. $\pi_{\text{minimax}} = \arg \min_{\pi : \mathcal{D} \rightarrow \mathcal{A}} \max_{z \in \mathcal{Z}_C} R(\pi, z)$ for some $\mathcal{Z}_C \subseteq \mathcal{Z}$.
- **Asymptotic optimality:** choosing π based on providing optimal performance in the limit of large data (noting that the true z can often be recovered from the data in this limit), e.g. zero-regret bandit strategies and estimators which converge at the optimal rate as given by the Cramér–Rao bound.

Another optimality criterion that is guaranteed to return an admissible policy is to choose the policy that optimises the so-called integrated risk,⁴

$$\bar{R}(\pi) = \mathbb{E}_{p(z)}[R(\pi, z)] \quad (53.6)$$

for some chosen distribution, $p(z)$. This, though, starts to blur the lines with the Bayesian approach as $p(z)$ is analogous to a prior in the Bayesian setup and one also arrives at the same mathematical integrated-risk formulation

⁴This is sometimes known as the Bayes risk. We stick with the term integrated risk because there is some inconsistency in the use of the term Bayes risk in the literature. For example, Robert [1405] used the term Bayes risk to refer to $\bar{R}(\pi_B)$, the integrated risk evaluated with the Bayes-optimal policy.

by taking an expectation of the Bayesian expected loss over possible observed data [123]. The interpretation is still arguably different.

Rather than trying to directly optimise some criterion, we can instead try to establish a policy that guarantees some required conditions regardless of the true value of z . For example, we might seek an estimator with provably consistent error bars.

Randomised decision-making

In the Bayesian setting, there is always a deterministic Bayes-optimal policy, $\pi_B : \mathcal{D} \rightarrow \mathcal{A}$, though it may not be the only Bayes-optimal policy [1050]. However, in frequentist settings, especially when using minimax optimality, it can be the case that the optimal policy is stochastic if either the action space, $\mathcal{A}$, or loss, ℓ , is not convex [123, Section 1.4]. For example, if $\mathcal{A} = \mathcal{Z} = \{0, 1\}$, $\ell(a, z) = \mathbb{I}(a \neq z)$ and we have no data, then $\max_{z \in \mathcal{Z}} R(\pi, z) = 1$ for any policy, π , that deterministically chooses a single action; but if our policy selects from $\mathcal{A}$ uniformly at random, then $\max_{z \in \mathcal{Z}} R(\pi, z) = 1/2$. In fact, this random policy turns out to be minimax-optimal.

As such, our earlier notation of decision rules as mappings $\mathcal{D} \mapsto \mathcal{A}$ was thus arguably a slight abuse of notation, and we should more carefully think of π as living in the space of stochastic mappings. For example, we can use a reparameterised decision rule, setting $a = \pi_r(d, \varepsilon)$ with $\varepsilon \sim p(\varepsilon)$, where $\varepsilon \in \mathbb{R}^n$ is some underlying randomness and we now have $\pi_r : \mathcal{D} \times \mathbb{R}^n \rightarrow \mathcal{A}$. The overall frequentist risk is then

$$R(\pi_r, z) = \mathbb{E}_{p(d; z)p(\varepsilon)}[\ell(\pi_r(d, \varepsilon), z)]. \quad (53.7)$$

We can, though, reformulate such problems back into cases of learning a decision rule that deterministically maps to a *distribution* over actions instead of a single action, by redefining our action space as a distribution over individual actions and/or redefining our loss as itself being an expectation, such that we now instead have $\pi : \mathcal{D} \rightarrow \mathcal{P}_{\mathcal{A}}$ and

$$R(\pi, z) = \mathbb{E}_{p(d; z)}[\bar{\ell}(\pi(d), z)] \quad (53.8)$$

$$\text{where } \bar{\ell}(\pi(d), z) = \mathbb{E}_{a \sim \pi(d)}[\ell(a, z)] = \mathbb{E}_{p(\varepsilon)}[\ell(\pi_r(d, \varepsilon), z)]. \quad (53.9)$$

To keep notation simple and maintain consistency with the Bayesian framework, in what follows we will thus assume the setting of a deterministic policy (i.e. decision rule), with the implicit assumption that the problem has been reformulated to the above in cases where it might be necessary for this to be deterministic to achieve optimality in our original action space.

53.2 Theoretically optimal experimental design

Having provided a solid foundation in decision theory, we now switch our focus to the specific problem of data-gathering, leading up to a formal definition of *Bayesian experimental design* [1050] and construction of the analogous frequentist setup.

53.2.1 Data-gathering policies

Whereas in 53.1 we considered choosing actions or policies based on some existing data, here we consider policies for the combined process of gathering data and then using it. In the Bayesian setup we will use d to represent the future data being gathered, with any existing data absorbed into our model and left implicit; and in the frequentist setup we will use d to represent the combined existing and future data.

Following Rossa et al. [1424], we start by breaking up the overall decision-making policy as $\pi := (\pi_d, \pi_a)$, where $\pi_d \in \Pi_d$ is the part of the policy that can influence the data-gathering, which we call the *design policy*, and $\pi_a : \mathcal{D} \rightarrow \mathcal{A}$ is the part of the policy associated with downstream actions once the data is collected (often known as the *terminal* decision problem), which we refer to as the *action policy*.⁵ Here the allowable set of data-gathering policies, Π_d , is dictated by the possible ways we can influence the data-gathering process. This is analogous to restrictions on the set of possible actions that can be performed in a more conventional post-data-gathering setting, and we can formally think of Π_d as being a space of “actions”.

It is important to note that the elements of Π_d themselves generally correspond to adaptive data-gathering policies that sequentially map from previously collected data to sequential decisions known as *designs*. As such, choosing an “action” does not generally itself correspond to choosing a specific design, but instead an autoregressive function that will be applied to make decisions during a sequential data-gathering process.

⁵Formally, π_a can also directly depend on π_d rather than just the data itself. We omit consideration of this in our notation for simplicity.

More concretely, assume there are T points at which we can choose designs and thus influence the data-gathering process.⁶ Further define $\xi_1, \xi_2, \dots, \xi_T$ as the corresponding series of designs with each $\xi_t \in \Xi_t$ and $o_t \in \mathcal{O}_t$ as the experimental outcomes observed between the t -th and $(t+1)$ -th decisions, such that overall data is given by $d = (d_1, d_2, \dots, d_T)$, where each $d_t = (\xi_t, o_t) \in \mathcal{D}_t = \Xi_t \times \mathcal{O}_t$. Note that the data is defined here as containing the design decisions for notational convenience (e.g. in active learning the design could be an input and it is natural to think of the data as input-output pairs). However, each design ξ_t itself provides no new information in isolation given the previously observed $d_{1:t-1} = (d_1, d_2, \dots, d_{t-1}) \in \mathcal{D}_{1:t-1} = \mathcal{D}_1 \times \mathcal{D}_2 \times \dots \times \mathcal{D}_{t-1}$.

We can now think of π_d as an autoregressive mapping from previous data to the next design decision, such that $\xi_t = \pi_d(d_{1:t-1})$ and

$$\Pi_d \in \Xi_1 \times \{f \mid f : \mathcal{D}_1 \rightarrow \Xi_2\} \times \dots \times \{f \mid f : \mathcal{D}_{1:t-1} \rightarrow \Xi_T\} \quad (53.10)$$

is a space formed by a collection of functional mappings from data *histories* to next design decisions. This setup is completely general in the sense that we utilise all information available at each step so that any optimal policy must conform to this form.

Analogously to in 53.1.2, in principle we can also use a stochastic design policy that instead maps from the previous data to a distribution of possible design decisions, such that $\pi_d(d_{1:t-1}) \in \mathcal{P}_{\Xi_t}$. Though this is sometimes needed in frequentist decision-making, and can also be helpful in practical training even when the optimal policy is deterministic [793], the optimal policy will still always be deterministic for all Bayesian and many frequentist settings. As in 53.1.2, we will thus stick to the notation of using a deterministic design policy, noting that this can still incorporate stochastic policies by viewing the design decisions as our choice of the distributions $p(\xi_t; d_{1:t-1})$, rather than the ξ_t themselves.

53.2.2 Bayesian experimental design

In the data-gathering setting we are uncertain about *both* z and the future data, d . Our Bayesian expected loss should thus be an expectation over both z and d . We also further have the restriction that our marginal beliefs on z should not depend on π_d , as our beliefs about the state of the world should not depend on the *future* policy we plan to make decisions with respect to. This then directly leads to the ActionBED formulation of the Bayesian future loss [1424]

$$L_B(\pi) = \mathbb{E}_{p(z, d; \pi_d)}[\ell(\pi_a(d), z)] = \mathbb{E}_{p(z)p(d|z; \pi_d)}[\ell(\pi_a(d), z)]. \quad (53.11)$$

Though formally the loss (and downstream-action policy) here can also depend on π_d and d (e.g. because different experimental choices incur different financial costs or to account for early-stopping rules), we will omit such consideration here in the interest of simplicity, referring the reader to Rainforth et al. [1370] for a more formal treatment of the fully general case. Note that we may, of course, also have access to historic data before d is gathered, but as per standard Bayesian decision rules, this influences our optimal decisions only through the beliefs on z , so we can omit direct consideration of this in our notation without loss of generality, as we simply update our definition of the prior, $p(z)$.

An overall Bayes-optimal policy is now a global minimizer for $L_B(\pi)$, that is $\pi^B = \arg \min_{\pi \in \Pi_d \times \Pi_a} L_B(\pi)$, with the Bayes-optimal design policy then just the part of this overall optimal policy corresponding to our design decision-making. However, existing work on practical BED methods has rarely directly reasoned about an explicit downstream-action policy (with Rossa et al. [1424] being the main notable exception in the modern literature). Instead, we can note that the coherence of Bayesian decision-making means that we can instead achieve a Bayes-optimal policy by making decisions sequentially in a way that is Bayes-optimal given the data seen so far and accounting for future decision-making through the loss we use at the current step. Namely, we can start with the optimal decision-making policy that would be carried out once data is observed, as per 53.3, then step backwards through the system, making each decision based on the knowledge that our future decisions will also be made Bayes-optimally. This then leads to the more classical definition for a Bayes-optimal design policy (see Rossa et al. [1424] for a proof of its equivalence to the minimiser of $L_B(\pi)$ discussed above).

Definition 53.6 (Bayes-optimal experimental-design policy; [254, 376, 584, 792, 799, 1050, 1364, 1424]). *A policy, π_d^B , for controlling a data-gathering process is known as a Bayes-optimal experimental-design policy for a given*

⁶Note that T does not necessarily need to itself be fixed, e.g. due to early-stopping rules. In such cases we can think of simply allowing “stop” as a design decision. We omit such consideration from our notation for simplicity, but note that none of our results or derivations rely on the assumption of a fixed T , with only the definition of Π_d itself needing updating in such cases.

belief-loss pairing, $(p(z)p(d|z; \pi_d), \ell)$, within an allowable space of possible policies, Π_d , if

$$\pi_d^B = \arg \min_{\pi_d \in \Pi_d} \mathbb{E}_{p(d; \pi_d)} \left[\min_{a \in \mathcal{A}} \mathbb{E}_{p(z|d)} [\ell(a, z)] \right], \quad (53.12)$$

where $p(z)$ represents our prior belief distribution over world states, $p(d|z; \pi_d)$ represents the conditional belief states over data given states for a particular data-gathering policy π_d , $\ell : \mathcal{A} \times \mathcal{Z} \rightarrow \mathbb{R}$ is a loss function that represents preferences of combined realisations of actions and states, $p(d; \pi_d) = \int p(z)p(d|z; \pi_d)dz$ is the implied marginal data distribution, and $p(z|d) = p(z)p(d|z; \pi_d)/p(d; \pi_d)$ is the implied posterior.

We can further now formally define *Bayesian experimental design* (BED) as being the study of data-gathering strategies that target a Bayes-optimal experimental-design policy or some approximation thereof.

A subtlety within this definition was that we deliberately wrote the posterior as $p(z|d)$ and not $p(z|d; \pi_d)$. This is because it turns out that the posterior is always independent of design policy given the observed data (noting that the data itself contains the design decisions) as shown by [1324], with the underlying intuition following from classical results on Bayesian ignorability and the likelihood principle [1432]. As such, the mechanisms for approximating posteriors can, if desired, be tackled independently from the problem of learning the design policy, e.g. by learning amortised-inference artefacts [794] or marginal score terms [1324].

From losses to uncertainties

While 53.6 reasons directly on the optimality of our policy in terms of the loss on terminal actions, Rossa et al. [1424] recently showed, based on earlier work [156, 376, 396, 677], that there is interestingly an alternative interpretation of Bayes-optimal design policies that only reasons about a proper scoring rule on our posterior beliefs, as demonstrated by the following result, which in turn we will reformulate to show that BED can be thought of in terms of minimising expected posterior uncertainty.

Theorem 53.1 ([1424]). *All Bayes-optimal experimental-design policies can be represented as minimisers of expected proper scoring rules on the posterior, and all minimisers of any expected scoring rule on the posterior are Bayes-optimal experimental-design policies for one or more belief-loss pairings.*

Namely, if $s : \mathcal{P}_z \times \mathcal{Z} \rightarrow \mathbb{R}$ is a proper scoring rule on $\mathcal{P}_z$ then

$$\pi_d^* = \arg \min_{\pi_d \in \Pi_d} \mathbb{E}_{p(z)p(d|z; \pi_d)} [s(p_z(\cdot|d), z)] \quad (53.13)$$

is a Bayes-optimal experimental-design policy for the belief model $p(z)p(d|z; \pi_d)$ and any loss function, $\ell \in \mathcal{L}$, where

$$\mathcal{L} = \{ \ell : \ell(\tilde{\pi}_a^\ell(q_z), z) = s(q_z, z) \ \forall q_z \in \mathcal{P}_z, \pi_d \in \Pi_d, d \in \mathcal{D}, z \in \mathcal{Z} \} \quad (53.14)$$

and

$$\tilde{\pi}_a^\ell(q_z) = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{z \sim q_z} [\ell(a, z)]. \quad (53.15)$$

Furthermore, $\mathcal{L}$ is never an empty set, and any integrable and measurable ℓ produces a corresponding s that is a proper scoring rule (though not necessarily a strictly proper scoring rule) given by

$$s(q_z, z) = \ell(\tilde{\pi}_a^\ell(q_z), z). \quad (53.16)$$

This is a somewhat profound result: it tells us that to conduct Bayes-optimal decision-making in the data-gathering setting, we do not need to explicitly account for downstream actions and a terminal loss if we are able to instead define an appropriate proper scoring rule on our future posterior distribution that also encodes our downstream preferences. Which of these is easier to faithfully specify will generally vary between problems.

This result becomes even more interesting when we realise that an expectation of such a proper scoring rule can be interpreted as an uncertainty measure, leading to uncertainty-based notions of Bayes-optimal design policies. In particular, given some distribution, $q(z)$, and some scoring rule, s , we can formally think of

$$h_s[q(z)] = \mathbb{E}_{q(z)} [s(q(\cdot), z)] \quad (53.17)$$

as defining a measure of uncertainty in $z \sim q(z)$ [376, 396]. In fact, this is arguably *the* definition of a Bayesian uncertainty measure; the choice of s here is not arbitrary but directly dictated by our downstream preferences via the corresponding loss function [1370].⁷ The resulting h_s is also known as a *generalised entropy* [677].

In turn, we can use this to define experimental-design objectives by considering the expected posterior uncertainty using such a measure:

$$\text{EPU}_s(\pi_d) = \mathbb{E}_{p(d;\pi_d)}[h_s[p(z|d)]] = \mathbb{E}_{p(z)p(d|z;\pi_d)}[s(p_z(\cdot|d), z)]. \quad (53.18)$$

Using 53.1, *all* BED objectives can be written in this way. As such, we can think of the set of h_s of the form 53.17 as being a *completely general notion of uncertainty* in a Bayesian-decision-theory context, in the sense that uncertainty measures that cannot be written as a particular h_s will not lead to Bayes-optimal decision-making.⁸

In practice, BED problems are often instead represented as *maximising* an expected uncertainty reduction from prior to posterior [156]:

$$\text{EUR}_s(\pi_d) = \mathbb{E}_{p(d;\pi_d)}[h_s[p(z)] - h_s[p(z|d)]] \quad (53.19)$$

$$= \mathbb{E}_{p(z)p(d|z;\pi_d)}[s(p_z(\cdot), z) - s(p_z(\cdot|d), z)]. \quad (53.20)$$

In the same way that h_s can be thought of as defining a generalised entropy, we can here think of the term $h_s[p(z)] - h_s[p(z|d)]$ as a *generalised information* (or generalised information gain) provided by the data. If the loss does not directly depend on π_d or d , then maximising such expected uncertainty reductions is exactly equivalent to the formal BED setup, but an extra cost term is required when it does.

We have now established that, alongside 53.6, we can formally think of BED approaches as equivalently minimising expected future losses of the form given in 53.11, minimising expected posterior uncertainties of the form given in 53.18, or maximising expected uncertainty reductions of the form given in 53.19 (with appropriate adjustment for costs if our loss depends directly on π_d and/or d). Using the terminology of h_s being a generalised entropy, we can also think of this last construction as maximising a *generalised expected information gain* (generalised EIG; [396]), noting that a gain in information is formally defined through a reduction in entropy.

53.2.3 Frequentist experimental design

Mathematically, it is straightforward to incorporate experimental design into the frequentist-decision-theory setup, as we simply use the fact that the data-generating procedure depends on the experimental-design policy. Namely, the frequentist risk becomes

$$R(\pi, z) = \mathbb{E}_{p(d;z,\pi_d)}[\ell(\pi_a(d), z)], \quad (53.21)$$

where, as with the Bayesian setup, $\pi = (\pi_d, \pi_a)$ with $\pi_d \in \Pi_d$ and $\pi_a : \mathcal{D} \rightarrow \mathcal{A}$ (and Π_d as per 53.10). Note here that the expectation in 53.21 is always over *both* past and future data: unlike in the Bayesian setup, we continue to reason about alternative data we might have previously seen, rather than taking it as fixed.

As with other frequentist decision-making, the optimal data-gathering policy will now depend on the optimality criterion used and in particular how we choose z . Unlike in the Bayesian setup, we are further not tied down to a particular mechanism for choosing π_a : frequentist systems do not need to be coherent and so we do not need to assume we will act Bayes-optimally in the future. For example, it is common to assume that we will later carry out maximum-likelihood estimation, motivating the use of strategies based around Fisher information matrices due to their link to the asymptotic maximum-likelihood-estimator distribution and the Cramér-Rao bound [342, 1379].

More broadly, the differences between the Bayesian and frequentist setups are more subtle in the data-gathering setting than in data analysis because we do not necessarily already have access to any data. Properly going into this is beyond the scope of this chapter; we refer to Rainforth et al. [1370] for more extensive discussion on the topic.

⁷More precisely, this can be viewed as the “uncentered” measure of Bayesian uncertainty, as it does not subtract the expected loss that would be incurred under perfect information. However, as this perfect-information loss is independent of any actions we actually take, it can generally be ignored for decision-making. See [1370] for details.

⁸Note that the interpretation here gets a little more complicated when ℓ , and by extension s , can depend on π_d and/or d , as here our design policy and/or data will change the form of the generalised entropy itself. In such cases, it arguably does not make sense to think about the expected future loss as an expected posterior uncertainty, as the “uncertainty measure” is itself being optimised in the process. An example of when this can occur is if there are costs associated with our data gathering, or if we want to control for particular properties of the data itself like privacy or fairness [726]. See Rainforth et al. [1370] for full discussion of this more complex case.

Scoring rule $s(q(\cdot), z)$	Uncertainty $h_s[q(z)]$	Expected uncertainty reduction $h_s[p(z)] - \mathbb{E}_{p(d; \pi_d)}[h_s[p(z d; \pi_d)]]$
Log loss $-\log q(z)$	Shannon entropy $H[q(z)]$	Expected information gain $\mathbb{E}_{p(z)p(d z; \pi_d)} \left[\log \frac{p(d z; \pi_d)}{\mathbb{E}_{p(z)}[p(d z; \pi_d)]} \right]$
Squared distance to mean $\ z - \mathbb{E}_{q(z)}[z]\ _2^2$	Trace of covariance $\text{Tr}(\text{Cov}_{q(z)}[z])$	Expected total variance reduction $\text{Tr}(\text{Cov}_{p(z)}[z] - \mathbb{E}_{p(d; \pi_d)}[\text{Cov}_{p(z d)}[z]])$
Hyvärinen [807] score $2\Delta_z \log q(z) + \left\ \frac{\partial \log q(z)}{\partial z} \right\ _2^2$	Fisher “entropy” $\mathbb{E}_{q(z)} \left[- \left\ \frac{\partial \log q(z)}{\partial z} \right\ _2^2 \right]$	Expected trace of Fisher information $\mathbb{E}_{p(z)} \left[\text{Tr} \left(\mathbb{E}_{p(d z; \pi_d)} \left[- \frac{\partial^2 \log p(d z; \pi_d)}{\partial z \partial z^\top} \right] \right) \right]$

Table 53.1: Common examples of proper scoring rules, their corresponding uncertainty measures, and the resulting BED objectives that maximise an expected uncertainty reduction. See Walker [1697] and Prangle et al. [1344] for proof that the Hyvärinen score (in which Δ_z denotes a Laplace operator) yields the expected trace of the Fisher information.

53.3 Practical experimental design

Building on the decision-theoretic foundations of experimental design from both a Bayesian and a frequentist perspective, we now turn our attention to how these frameworks are in turn used to (adaptively) optimise experiments. Properly covering the large space of existing approaches here is inevitably not feasible, so we instead focus on two classes of approaches that together cover the vast majority of methods used in practice: those based on expected uncertainty reduction, including information-theoretic approaches; and those based on Fisher information matrices. These respectively correspond to the most common approaches within the Bayesian and frequentist frameworks.

53.3.1 Expected-uncertainty-reduction approaches

When working within the Bayesian framework, it has historically been rare for practical methods to explicitly reason about downstream actions after the data has been gathered, with almost all methods instead implicitly relying on 53.1 to express criteria in terms of expected scoring rules on the posterior. In particular, most methods used in practice formulate BED problems in terms of maximising an *expected uncertainty reduction* (EUR) from the prior to the posterior (53.2.2); some prominent examples are presented in 53.1. Here we discuss some important features of EUR approaches in general before paying closer attention to the particular case of maximising expected information gain.

Key considerations and practicalities

Indexing between different BED objectives simply comes down to different choices of our model and scoring rule (or downstream loss in the cases this is done directly). Fundamentally, the success of any BED approach thus relies on three things:

- (a) the faithfulness of the choice of uncertainty measure, scoring rule or loss to our actual preferences;
- (b) the effectiveness of our model in capturing both the true data-generating process (remembering $p(d; \pi_d)$ represents our beliefs about possible data that might be generated, not the ground-truth process) and our beliefs and uncertainty (where the way we measure this uncertainty is itself implied from the choice of scoring rule using h_s); and
- (c) our ability to practically optimise the BED objective and thus find a suitable data-gathering policy, π_d .

It is difficult to talk about how to achieve (a) and (b) in a fully general sense, as they are inherently about the accuracy to which our problem construction truly conveys the real world and our preferences, while these considerations are also often wrapped up with the practical considerations, (c). It is thus beyond the scope of this chapter to really delve into these fully, but we note that Shannon entropy has become almost the canonical measure of uncertainty, as discussed in 53.3.1. We also highlight the clear link to Bayesian deep learning, in that our success is rooted in our ability to capture uncertainty.

Much of the BED literature, especially in recent times, has focused instead on (c). EPU and EURs are inherently doubly intractable, making them extremely difficult to estimate and optimise [1367, 1369]. This can become particularly challenging when dealing with adaptive sequential experimentation, where approximations are typically made to the pure BED framework to make design strategies practical. As we will see in 53.3.4, various methods have been proposed to try and circumvent, or at least mitigate, these computational issues.

An important point to note here is that methods should *not* be considered Bayesian just because they incorporate (predictive) uncertainty, even if that uncertainty is derived from a Bayesian model. The key feature of BED approaches is reasoning about expected *future* losses or uncertainties, not the immediate predictive uncertainty. For example, gathering data in active learning by choosing inputs with the highest predictive uncertainty has no direct justification from Bayesian decision theory and is implicitly based on the heuristic assumption that if one is more uncertain about the outcome of an experiment then it will likely provide more useful data.

This distinction between Bayesian decision-making and Bayesian models is an important one: the problem of experimental design is fundamentally one of decision-making and the use of Bayesian models stems from the underlying decision theory (and is specific to the sequential decision-making setting) rather than the other way around.

Furthermore, there are no universal measures for how effectively a model captures uncertainty. So when justifying models like BNNs, this must be rooted in the tasks we wish to conduct and the decisions we want to make to be meaningful [1329, 1501]. Indeed, an important result for all BED approaches is that we only need to consider uncertainty in variables that directly influence our downstream loss [1370]. Thus, when defining BED procedures through uncertainty measures or score functions, we should make sure these are always in the space of the minimal set of variables that we expect any hypothetical future losses to directly depend on. For example, if we are interested in prediction then we should define a proper scoring rule on the posterior predictive, not the model parameters, and if we care about estimation of some particular quantities then we should define a scoring rule measuring the error in estimating those precise quantities rather than on some preimage or mapping of them.

Expected information gain

By far the most common form of expected uncertainty reduction in the BED literature is the *expected information gain* (EIG) of the experiment, specifically the expected reduction in Shannon entropy from the prior to the posterior [129, 1049, 1050]. This exactly conforms to the formal BED setup, simply corresponding to the expected uncertainty reduction that results from using log loss on beliefs about some variables of interest, ψ , as our associated proper scoring rule, yielding the objective

$$\text{EIG}_\psi(\pi_d) = \mathbb{E}_{p(d; \pi_d)}[\text{H}[p(\psi)] - \text{H}[p(\psi|d)]] = \mathbb{E}_{p(\psi)p(d|\psi; \pi_d)} \left[\log \frac{p(\psi|d)}{p(\psi)} \right] \quad (53.22)$$

where H denotes Shannon entropy [1499]. Note here that we have deliberately formulated the EIG in terms of a particular set of variables of interest instead (which are implicitly some pushforward of the world state, z) because of the arguments presented in 53.3.1 and because the ability to target the EIG in different quantities often itself provides alternative experimental-design strategies (e.g. the BALD vs EPIG acquisition objectives covered in 53.4.2 respectively target EIG in model parameters and predictions).

The EIG admits a wide variety of different useful equivalent forms for any given choice of design policy, such as the *mutual information* between ψ and the data, along with the following equivalent formulations:

$$\text{EIG}_\psi(\pi_d) = \mathbb{E}_{p(\psi)p(d|\psi; \pi_d)} \left[\log \frac{p(d|\psi; \pi_d)}{p(d; \pi_d)} \right] \quad (53.23)$$

$$= \mathbb{E}_{p(\psi)}[\text{H}[p(d; \pi_d)] - \text{H}[p(d|\psi; \pi_d)]] \quad (53.24)$$

$$= \mathbb{E}_{p(d; \pi_d)}[\text{KL}[p(\psi|d) \parallel p(\psi)]] \quad (53.25)$$

$$= \text{KL}[p(\psi)p(d|\psi; \pi_d) \parallel p(\psi)p(d; \pi_d)] \quad (53.26)$$

where KL denotes the KL divergence [945]. It has been justified as an experimental-design objective in a wide variety of different ways using both Bayesian decision theory and information theory (e.g. [129, 396, 790, 1049, 1050, 1369]). Many of these different interpretations are important in different contexts, particularly when it comes to the practical estimation and optimisation of the EIG. For example, because the EIG is equivalent to a mutual information for any given design policy, we can utilise the symmetry of mutual informations to avoid ever having to directly work with posterior distributions, using the ratio of the likelihood to the marginal likelihood as per 53.23 instead. In turn, this is critical in applications like active learning (where ψ can be the parameters

of a large model but the label we acquire could be a single categorical variable) as discussed later, but also other large-scale BED settings as well [300].

While its overwhelming popularity in the experimental-design setting likely stems, in part, from simply being the best-known approach that formally follows the BED framework (to the extent that the two have often historically been seen as synonymous), it does also have many appealing properties that provide some compelling justification for this status. For example, it is unique in being invariant to reparametrisations and even injective mappings of ψ , and is also unique in being a *strictly* proper scoring rule that only depends on the probability (density) of events that actually occur rather than hypothetical alternatives; see Rainforth et al. [1370] for details.

These properties make the EIG a very appealing choice as a default strategy for BED when we are unsure exactly how our data will be used. In particular, while VNM utility theory tells us that, in principle, a loss exists that encapsulates our preferences over downstream consequences whether we know exactly how the data will be used or not (with this loss implicitly encapsulating possible ways the data *could* be used if we do not yet know), in practice it can be difficult or impossible to properly write these down in the form of a concrete function. The EIG is often an attractive proposition in such settings, though care should still be taken about *which* variables we target the EIG in.

53.3.2 Fisher-information and estimator-sharpness approaches

The classical, frequentist approach to experimental design has a long and extensive history dating back to before Bayesian (and most frequentist) decision theory had been developed. Early seminal work in the area includes that of Fisher [527], Kiefer [886], Smith [1525] and Wald [1692].⁹

We will focus on the predominant approach of choosing designs according to minimising invariants of the *Fisher information matrix* (e.g. determinant or trace of inverse), defined as [524, 527]

$$F(\pi_d, z) = \mathbb{E}_{p(d; z, \pi_d)} \left[\left(\frac{\partial}{\partial z} \log p(d; z, \pi_d) \right) \left(\frac{\partial}{\partial z} \log p(d; z, \pi_d) \right)^\top \right] \quad (53.27)$$

$$= \mathbb{E}_{p(d; z, \pi_d)} \left[-\frac{\partial^2}{\partial z \partial z^\top} \log p(d; z, \pi_d) \right]. \quad (53.28)$$

The Fisher information represents the curvature of the log likelihood around a particular value, z (whose value must generally be chosen using some optimality criterion as per standard frequentist decision theory). We can thus think of it as a *local* measurement of the *sharpness* of the likelihood around z : if the likelihood is very tightly peaked, then small variations in z dramatically change the likelihood of the data, implying that the data is (locally) highly informative for learning about z . In the frequentist literature it is typically described as measuring how much information the data provides about the parameters.

This notion of information is different to the Shannon information and generalised informations we have seen in the BED literature, but does share some similar properties, such as the chain rule of expected uncertainty reductions. We can think of FIMs as measuring a set of uncertainty reductions given by the Hessian of the corresponding densities [1370], which have the unusual, and from a frequentist perspective highly beneficial, property of being independent of the prior when we look at the difference in score between prior and posterior. However, we should also note that it is just one *possible* choice of uncertainty, rather than being appropriate for all problems. In fact, it can lead to various pathologies when used in practice, such as the fact it is generally optimised by *non-adaptive* design strategies unless adaptation is forced from the optimality criterion itself [1370], in turn sometimes leading to catastrophic failures when used as an adaptive design or active-learning strategy, such as never reducing uncertainty in certain dimensions of z .

Because $F(\pi_d, z)$ is a matrix instead of a scalar, it is not itself a suitable objective for optimal experimental design, and we must instead optimise some invariant of it. Different choices of the chosen invariant are typically known as the *alphabetic criteria*. Below we highlight some common choices.

- **A-optimality**: minimising the trace of the inverse of the FIM (the A stands for “average”). It can be viewed as minimising the asymptotic sum of the variances of different dimensions of z . Though computationally convenient, it can be quite a brittle objective in practice, as the optimal designs change under reparameterisation or even just rescaling of the dimensions of z (if we scale a dimension of z by a factor of 10, that dimension will have 10 times less influence on the objective). It also fails to account for any reduction in information that comes from co-dependence between different dimensions of z .

⁹Covering the long history in this area is beyond the scope of this chapter. We refer the reader to Atkinson et al. [59] and Pukelsheim [1347] for comprehensive textbook references.

- **c-optimality**: minimising $v^\top F(\pi_d, z)^{-1}v$ for some projection vector v . Sometimes this is extended to the trace of $M^\top F(\pi_d, z)^{-1}M$ for a matrix M (sometimes known as L-optimality) or minimising for a worst case projection vector v (sometimes known as G-optimality). Prangle et al. [1344] further provide a formal link to working with worst-case matrix projections to D-optimality below.
- **D-optimality**: maximising the determinant of the FIM. This can be viewed as maximising the “volume” of the FIM and equivalently can be viewed as maximising the geometric mean of the FIM eigenvalues or minimising the asymptotic generalised variance (i.e. the determinant of the covariance) of the maximum-likelihood estimator for z . In general, the D-optimal designs do not change under smooth one-to-one reparameterisation of z .
- **E-optimality**: maximising the minimum eigenvalue of the FIM. This can be useful as a robust criterion where we need to make sure all components of z are estimated well; it can be thought of as minimising the asymptotic largest principal-axis variance of the maximum-likelihood estimator for z .

Of these criteria, A-optimality and D-optimality are generally the most commonly used, with the former typically favoured for computational reasons and the latter because of its reparameterisation invariance and incorporation of codependency effects.

As with all frequentist methods, we also need to have some kind of optimality criterion for dealing with the fact that z is unknown. Bizarrely, this problem is often somewhat ignored in the frequentist-experimental-design literature, especially in older papers. The primary reason for this is that if one uses a linear Gaussian model of the form $d|z; \xi \sim \mathcal{N}(\xi z, \sigma^2)$ for some fixed variance, σ^2 , then the FIM turns out to be independent of z and simply equal to $\xi \xi^\top / \sigma^2$. Thus, for the linear models that have dominated much of classical statistics, this problem conveniently disappears. In the modern world this is, of course, far less often the case and we must indeed come up with a mechanism to choose z . In principle, any admissible optimality criterion can be used, but perhaps the most common is simply to use a point estimate for z , typically the current maximum-likelihood estimate given previous data [290]. This assumption of a particular value for the precise quantity we are gathering data to estimate is arguably somewhat unsatisfying. More complex procedures have been proposed as well, such as minimax approaches [419].

53.3.3 Adaptive design

Optimal experimental design often requires us to be *adaptive*, utilising information gathered from previous data to guide our design decisions and learn on the fly. Namely, following 53.2.1, we can use the data gathered so far, $d_{1:t-1}$, to help inform each sequential design decision, ξ_t , where t indexes the design decisions we must make as part of the experimental design process. This adaptive aspect of optimal experimental design has been implicit in everything we have covered, through the notion of a design policy, π_d , with design decisions explicitly framed as mappings from previously observed data.

Traditional myopic adaptive design

The idea of training an *explicit* design policy is relatively new, with the concept of a design policy first introduced by Huan and Marzouk [789], and later refined into the form presented in 53.2.1 and popularised by Foster et al. [552] and Ivanova et al. [823], with the concept of a downstream action policy in turn introduced by [1424]. Prior to this work, the standard approach to adaptive design had been to follow an *implicit* policy, in which each individual design decision, ξ_t , is chosen by directly optimising some experimental-design objective that incorporates the data gathered so far, rather than following a pre-determined decision rule. In the BED setting, this objective has typically been taken to be the *incremental* EIG in the target variables ψ ,

$$\text{EIG}_{\psi|d_{1:t-1}}(\xi_t) = \mathbb{E}_{p(d_t|d_{1:t-1}, \xi_t)} [\text{H}[p(\psi|d_{1:t-1})] - \text{H}[p(\psi|d_{1:t})]], \quad (53.29)$$

which is equivalent to a standard EIG on the next experiment step, with a prior given by our current posterior beliefs given the data observed so far, $p(\psi|d_{1:t-1})$. Frequentist and active-learning approaches often follow a similar procedure with non-Bayesian updates to incorporate new data (e.g. using point estimates based on maximum-likelihood estimation in frequentist settings, or re-running a training procedure on all the current data in active-learning settings). In what follows, we will mostly focus on the BED setting, noting that adaptation has historically been a bigger focus for the Bayesian literature than for the frequentist literature.

An important result shown by Foster et al. [552] and then refined by Ivanova et al. [823] is that the expected sum of the incremental EIGs is equal to the total EIG across all experiment iterations:

$$\mathbb{E}_{p(d_{1:T-1}; \pi_d)} \left[\sum_{t=1}^T \text{EIG}_{\psi|d_{1:t-1}}(\xi_t) \right] = \mathbb{E}_{p(d_{1:T}; \pi_d)} [\text{H}[p(\psi)] - \text{H}[p(\psi|d_{1:T})]] = \text{EIG}_{\psi}(\pi_d), \quad (53.30)$$

where π_d can be either an explicit or implicit policy. This result turns out not to be specific to EIGs, but holds for EURs more generally [1370]. The implication of this is that the best overall policy is the one that maximises the average of the individual incremental EURs at each step. There is thus an underlying *dense* reward (in the reinforcement-learning sense) that can help provide signal for individual design decisions and provides some justification for this incremental approach.

However, this does not mean that this incremental approach will generally be optimal: it is making *myopic* (or “greedy”) decisions that do not account for the fact that the design decision made at one iteration may influence not just the EUR at the next experiment iteration, but also future EURs as well. For example, imagine a scenario where we have one-dimensional designs in the space $[0, 1]$ and our loss is such that the EPU is minimised when the designs are as evenly spaced as possible. If we have two designs and plan ahead about where to place them, the optimal designs will be at $1/3$ and $2/3$. If we follow a myopic approach, though, the first design will be placed at $1/2$ and the second at either $1/4$ or $3/4$, leading to a sub-optimal final spacing. In the language of reinforcement learning, we are essentially neglecting the fact that the value function typically varies between different design decisions.

Explicit-policy approaches

In addition to the theoretical suboptimality from myopic decision-making, the traditional adaptive design approach has two critical weaknesses: (a) all computation must be performed *during* the experiment and (b) for all Bayesian and some frequentist approaches, we need to perform model updates between experiment iterations. Both of these can be extremely computationally restrictive in practical problems, where design decisions must typically be made in real time to be useful.

To circumvent these issues, many modern BED approaches instead look to learn an explicit policy, which can then be directly evaluated during the experiment to make design decisions quickly. One can view such approaches as amortising the overall experimental-design problem, performing all required computation upfront rather than during the experiment. While still a groundbreaking development, the first policy-based approach, introduced by Huan and Marzouk [789], was not well suited in this regard as it used the posterior beliefs as the “state” within a reinforcement-learning problem construction, meaning that inference (and thus substantial computation) was still required at deployment time.

The key breakthrough for solving these problems was provided by Foster et al. [552], whose deep adaptive design (DAD) approach instead constructed the design policy as a deep neural network that directly maps from the current data itself, $d_{1:t-1}$, to the next design decision, ξ_t , as per the framework presented in 53.2.1. This then allowed the design decisions to be made in a matter of milliseconds through a simple forward pass of their learned *design network*. They further showed that this policy can be trained using end-to-end stochastic gradients through the full experiment rollout, training directly on (bounds or estimates of) the total EIG, $\text{EIG}_{\psi}(\pi_d)$. That is, DAD trains the design network by generating a full dataset using the current policy, estimating the information gain provided by that dataset, then backpropagating through this whole process to update the policy.

Interestingly, DAD showed significant performance improvements over the traditional myopic approach in design quality, not just computational cost at deployment. Part of the reason for this is in allowing non-myopic policies that plan ahead to be learnt. A second, more subtle factor is that it avoids the need to explicitly perform inference between experiment steps; this is critical because inaccuracies in this inference transpire to be one of the main sources of error in previous approaches, and learning direct mappings from previous data to next design decision is sometimes easier than the posterior-inference problem itself. It was also found that the cost of training effective design networks was often comparable to running a *single* rollout of the traditional myopic framework, such that this approach can still be very effective and practical even when fast deployment is not a requirement. Consequently, virtually all current state-of-the-art adaptive BED methods now rely on some form of explicit policy learning.

A wide variety of work has since directly built on DAD’s explicit policy-learning approach. For example, Ivanova et al. [823] extended the framework to work on implicit models when the likelihood density is not available by introducing a learned critic network, and Hedman et al. [725] showed that periodically finetuning the policy during the experiment using collected data can provide benefits.

Following Huan and Marzouk [789], other work has formulated the policy-learning problem through the lens of reinforcement learning [168, 1040], while using data collected as the state so far, as in DAD, to ensure computational efficiency at deployment. Huang et al. [792] took this further by allowing for different downstream utilities than EIG; Huang et al. [793] built on this to simultaneously learn amortised predictive distributions that can help support the training of the policy. Huang et al. [794] showed that neither the raw data nor an explicit posterior approximation are typically optimal for representing the current state as the input to the policy network, showing that significant empirical gains can be achieved by using internal representations of amortised-inference foundation models, which can help capture invariances in the posterior without introducing the error that originates from an explicit approximation.

Most recently, Rossa et al. [1424] proposed directly targeting expected future losses instead of expected uncertainty reductions and co-training an explicit downstream action policy alongside the explicit design policy, and showed that this approach can simplify training while providing improvements in both customisability and performance.

53.3.4 Estimation of BED objectives

A property shared by virtually all BED objectives is that they are very difficult to estimate and optimise, because we are inherently reasoning about possible future beliefs given data, rather than just our current beliefs. In particular, unless the posterior has a known analytic form, the general BED formulation as an expected scoring rule of the posterior given in 53.18, and by extension the expected-uncertainty-reduction formulation of 53.19, are *nested expectations* that are *doubly intractable* to estimate except in special cases [905, 1367], with the inner expectation corresponding to the unknown normalising constant of the posterior. That is, we are trying to optimise an expectation of a nonlinear function of an expectation (noting that non-constant proper scoring rules must be nonlinear), which cannot generally be done directly with standard techniques like Monte Carlo and stochastic gradient descent, and instead requires more complex and costly strategies like nested Monte Carlo [1367] or multi-level Monte Carlo [628].

Extensive research has been conducted into how to best carry out this estimation in the specific case of using the EIG as the objective, with such work forming a prominent part of the BED literature in recent years. Examples of approaches that have been proposed include nested Monte Carlo [330, 1219, 1365, 1367, 1444], variational and (amortised) density-approximation methods [93, 550–552, 793, 1026], multi-level Monte Carlo [454, 643], Laplace approximations [103, 247, 1073], score-based approximations [1324], and other approximate sampling-based strategies [38, 228, 818, 819, 821, 1459]. It is notable that the problem becomes even more challenging when working with *implicit* models, wherein we can sample from our likelihood but not evaluate its density [823, 903]. Covering these properly is beyond the scope of the current work; we refer the reader to Huan et al. [790] and Rainforth et al. [1369] for comprehensive recent reviews on the topic.

Research on dealing with the required estimation for other losses has been more limited, but is starting to become a more active area of research. The most common traditional approach in such settings, which have been more commonly considered in the specific active-learning literature than the rest of the experimental-design literature, has been to rely on 53.1 to formulate the BED objective as an EUR where the scoring rule is directly specified rather than the loss itself, then calculate approximate posteriors (or retrain a model in a non-Bayesian way) each time new data is sampled [1429], thus treating it as a nested-inference problem [1366]. More modern work has built on this with amortisation, reweighting, exploiting an ensemble, or similar [792, 1218, 1589]. Various methods have also been proposed for specific other EUR objectives, such as expected variance reduction [320, 758].

Huang et al. [799] recently devised an approach for the setting where the loss is directly specified, noting that deriving the scoring rule from the underlying loss is itself typically intractable. Namely, they showed that whenever the loss takes the form of a weighted Bregman divergence, the optimisation over actions in 53.12 has an analytic solution where the optimal action is some kind of weighted posterior mean, such that the scoring rule associated with the loss can be derived without requiring a nested optimisation to be performed. In turn, this allowed them to derive a simple and practical (albeit expensive) nested Monte Carlo estimator that can be applied to any BED objective whose loss takes this form, which is itself quite general and incorporates many useful losses, including those that give rise to the EIG and expected variance reduction.

Finally, Rossa et al. [1424] recently suggested a new approach that can be applied for any downstream loss function, only requires optimisation of a singly intractable objective without any posterior approximations at all, and which is naturally implicit (i.e. it does not require access to model densities, only forward samples). This approach is notably straightforward: instead of relying on 53.1 to reformulate the problem into uncertainties, they simply introduce an explicit action network, analogous to the design network in deep adaptive design (53.3.3), and then directly minimise the expected future loss in the form of 53.11 using stochastic gradients. Their critical

insight is that, if we go back to this foundational formulation of what it means to be Bayes-optimal and jointly train design and action policies together, this yields a singly-intractable problem formulation that is naturally task-driven and avoids posterior calculations completely (even after the data has been gathered), thereby providing a much simpler and, at least in principle, more computationally friendly framework than dealing with EUR. In other words, the nested nature of BED objectives is actually an artefact of reformulating them into EPU or EURs, rather than inherent to the original problem formulation. Though the impact this work will have on the field remains to be seen, it suggests an interesting potential paradigm shift in how we think about BED, moving from the traditional idea of maximising EUR, to instead thinking directly about minimising expected future loss on final tasks of interest.

53.4 Active learning

So far we have used statistical decision theory to identify notions of optimality in experimental design, and we have discussed the translation of these into practical methods. Now we turn to active learning, which is often considered synonymous with experimental design but which we suggest can more usefully be interpreted as a particular form of experimental design focused on *accurate predictions* of outputs for inputs that are often unknown at the time of data-gathering. We highlight distinctive aspects of active learning, review a number of popular data-gathering methods, and discuss some key methodological considerations.¹⁰

53.4.1 Overview

Like other technical topics, active learning can be thought of as three things at once: “a problem, a class of solution methods that work well on the problem, and the field that studies this problem and its solution methods” [1579]. As a field it is home to much of the machine-learning field’s study of intelligent data-gathering, while also being effectively separate from the machine-learning subfield of reinforcement learning and the statistical subfield of experimental design. These fields overlap but it is useful to distinguish between them because of differences in their predominant problem definitions and solution methods, which we explore throughout this section.

Problem

Active learning as a problem class inherits the structure from 53.2. We seek a combination of design policy, π_d , and action policy, π_a , that minimises the expected value (under a Bayesian or frequentist formulation) of a loss function, ℓ , with our choice of π_d influencing the sequence of designs, $\xi_{1:T}$, and the new data, $d_{1:T}$, we observe.

Where the active-learning literature is most distinctive is in its explicit focus on accurately predicting the outputs, $y_{1:N} \in \mathcal{Y}^N$, corresponding to $N \geq 1$ target inputs, $x_{1:N} \in \mathcal{X}^N$, where the relationship between inputs and outputs is often very complex. Common prediction problems include image classification [569, 1489], text classification [1019, 1493], speech recognition [452, 1643] and molecular property-prediction [707, 1838]. In *inductive* problems $x_{1:N}$ is not known until after data-gathering, whereas in *transductive* problems it is known up front [1667].

Also notable is an emphasis on high-dimensional design spaces, Ξ_t , and data spaces, $\mathcal{D}_t$, where again we typically face a challenging prediction problem in mapping from designs to data. There are active-learning problems with low-dimensional spaces [60, 321, 322, 1106, 1107] and alternative experimental-design problems with higher-dimensional spaces [551, 903, 1849]. But the highest-dimensional spaces studied so far have overwhelmingly been within the active-learning literature. For example, it is common in active learning for ξ_t to represent a batch of images and d_t to represent those images combined with class labels [52, 54, 898, 900, 1489], such that Ξ_t is a continuous-valued vector space that can easily have millions of dimensions, and $\mathcal{D}_t$ is a discrete space of size exponential in the batch size.

While active learning is flexible in its notion of designs and data, for concreteness we will focus on a variant of that common image-classification case in which each design, $\xi_t \in \mathcal{X}^B$, is a batch of $B \geq 1$ inputs (not necessarily images) for labelling and each unit of data, $d_t \in \mathcal{Y}^B$, is a corresponding batch of input-label pairs (not necessarily with categorical labels), where these inputs and labels exist in the same spaces as the target inputs and outputs. Often there will be some initial data, d_0 , which can comprise just labelled data, just unlabelled data, or a mixture. It is typical in the active-learning literature for d_0 to contain a discrete set of unlabelled inputs known as a *pool* [1019, 1494], which we denote as $\mathcal{X}_{\text{pool}} \subseteq \mathcal{X}$. In the case of transductive learning, another form of unlabelled data known ahead of data-gathering is $x_{1:N}$.

Whereas handling high-dimensional design and data spaces is a practical difficulty that is increasingly addressable in other experimental-design settings [1459], the focus on challenging predictions stands to remain a distinctive

¹⁰See Rainforth et al. [1370] for a deeper decision-theoretic treatment of active learning.

feature of active learning as a problem class. In principle the scope of active learning could be expanded to include alternative goals, to the point of it being synonymous with experimental design. But maintaining a tighter scope—one that reflects the de-facto definition of active learning in the literature—allows us to more efficiently communicate substantive differences in emphasis between communities.

Models

The nature of active learning as a problem helps explain some salient features of active-learning methods in relation to alternative experimental-design methods. One such feature is how we construct models. The emphasis on accuracy in challenging prediction problems typically requires us to forgo conventional statistical-modelling approaches in favour of generic *predictive algorithms*, crossing a historical gap between machine learning and statistics [206] that is arguably only now starting to meaningfully reduce [538, 540]. These predictive algorithms often produce more accurate models while also posing relatively low computational costs, albeit with the notable caveats that they can struggle in the low-data regimes commonly associated with active learning [921] and that they generally do not produce coherent belief updates [501, 1329].

In some cases the models we construct will contain stochastic parameters, $\theta \sim p(\theta; d)$. These are often defined such that predictions are independent conditioned on θ :

$$p(y_{1:N}; x_{1:N}, d) = \mathbb{E}_{p(\theta; d)} \left[\prod_{n=1}^N p(y_n | \theta; x_n, d) \right]. \quad (53.31)$$

This setup encompasses most conventional Bayesian models (following standard arguments; [383]), for which $p(\theta; d) = p(\theta|d) \propto p(d, \theta)$ is an exact Bayesian posterior derived from an explicit joint model, $p(d, \theta)$, and $p(y_n | \theta; x_n, d) = p(y_n | \theta; x_n)$ is a fixed sampling distribution. It also includes ensemble models [426] and stochastic neural networks [1501].

Another factor in how we construct models is whether we have a mixture of labelled and unlabelled data. For example, if d_0 contains unlabelled inputs and $d_{1:t}$ contains input-label pairs then $p(\cdot; d_{0:t})$ could be a *semi-supervised* model that incorporates information from both the labelled and unlabelled data. Early work along these lines used a range of models, including ensemble [1643], Gaussian-mixture [1237], Gaussian-random-field [1879], n -gram [1396] and naive-Bayes [1156, 1217] models. More recent work has tended to focus on deep-learning models that combine an encoder and a prediction head, where the encoder is pretrained on unlabelled data and then either the whole model or just the prediction head is trained on labelled data [55, 155, 698, 1120, 1130, 1194, 1271, 1502, 1587, 1815, 1827].

In addition to a model of outputs given inputs, for some methods we need a model of the unknown target inputs. This could be a discrete distribution over the pool, $\mathcal{X}_{\text{pool}}$ (if it exists); a model trained on $\mathcal{X}_{\text{pool}}$; or an analytic distribution over the input space [154, 1156, 1429].

Data-gathering

Aside from the models and action policies they use, active-learning methods are distinctive in how they typically select designs, ξ_t , even though there are no restrictions on this in principle (i.e. within the problem definition). First, whereas in 53.2 we showed that optimal data-gathering needs to plan ahead based on full simulations of $(\xi_t, d_t)_{t=1}^T$, the vast majority of active-learning methods choose a single design at a time and treat each design decision as if it were the last. This *myopic* approach (matching that discussed in 53.3.3 except that design selection does not necessarily involve optimisation) reduces the difficulty of forecasting future data and the resulting model updates, which is particularly challenging in active learning due to the complex design–data relationships and flexible predictive algorithms involved. Nonmyopic approaches remain rare but are beginning to receive more attention; Huang et al. [794] recently presented a successful demonstration.

Second, even if ξ_t is an element of a continuous design space, Ξ_t , it is common to construct a discrete design space, $\Xi_{\text{pool}} = \mathcal{X}_{\text{pool}}^B$, based on a pool of unlabelled inputs, $\mathcal{X}_{\text{pool}}$, available (or generated) at the start of data-gathering (53.4.1), and then to select ξ_t from Ξ_{pool} . This contrasts with *query synthesis* [1494, 1652], in which ξ_t is generated from scratch. Pool-based design selection can help filter out pathological designs: if ξ_t represents an image for a human to label, we might want to avoid selecting images that are hard to label [972, 1459], and we can do that by using a pool of “natural” images.

Third, the selection of ξ_t from Ξ_{pool} does not always take the form of a straightforward optimisation. Many methods formulate an *acquisition objective* that can be evaluated on any candidate input batch, ξ , and in principle we can evaluate this objective on all $\xi \in \Xi_{\text{pool}}$ before selecting the ξ with the best value (lowest for some methods; highest for others). While this exact-optimisation approach is typical when ξ_t represents $B = 1$ input, it becomes

costly for selecting larger batches of $B > 1$ inputs. Some methods instead approximate the optimisation over Ξ_{pool} by greedily constructing ξ_t , namely choosing the best first input, the best second input given the chosen first input, and so on [52, 898, 1489]. Others use stochastic sampling of ξ_t , for example using a Boltzmann distribution derived from the acquisition objective [506, 835, 900, 1837], by top- B selection based on a two-sample Monte Carlo estimate of the objective [834], or by Thompson sampling [842]. There are also methods that do not formulate an objective at all, such as BADGE [54], a method we cover later.

53.4.2 Data-gathering methods

Here we present some of the most prominent data-gathering methods in the literature. Reflecting the dominance of myopic approaches, we consider selecting a single design, ξ_t , rather than a design policy, π_d . For notational simplicity we also default to not indexing by step, t , and leaving the dependence of models on the current data, $d_{0:t-1}$, implicit.

Information-based methods

The idea of maximising EIG (53.3.1) in active learning, which was pioneered by MacKay [1107] and later became known as *Bayesian active learning*, has been very influential within the field. Multiple forms of EIG have been proposed, each corresponding to an expected reduction in Shannon entropy in a chosen variable.

Parameter EIG MacKay [1107] studied four forms of EIG. The form that went on to receive the most attention is what he called the expected total information gain (ETIG), now widely known as the BALD score following work on “Bayesian active learning by disagreement” [784, 785]. The BALD score for a candidate design, ξ , is the expected information gain in a set of stochastic model parameters, $\theta \sim p(\theta)$, from observing the data, d , produced by ξ :

$$\text{BALD}(\xi) = \text{EIG}_{\theta}(\xi) \quad (53.32)$$

$$= \text{KL}[p(d, \theta; \xi) \parallel p(d; \xi)p(\theta)]. \quad (53.33)$$

MacKay [1107] proposed an estimator of this objective for regression with a stochastic neural network, Houlsby et al. [784, 785] an estimator for classification with a Gaussian-process model and Gal et al. [569] an estimator for classification with a stochastic neural network; the last of these was also concurrently proposed in the BED literature [1365, 1367]. Later Kirsch et al. [898] proposed BatchBALD, a greedy scheme for approximately optimising $\text{BALD}(\xi)$ over $\Xi_{\text{pool}} = \mathcal{X}_{\text{pool}}^B$ for $B > 1$.

Region-of-uncertainty sampling [60, 321] and query by committee [1496] can be understood as instances of BALD-based acquisition with a particular form of model whose predictions of d are deterministic given ξ and θ [559]. The former randomly samples (not necessarily uniformly) from the set of candidate inputs with nonzero BALD score, while the latter maximises BALD.

Transductive predictive EIGs MacKay [1107] also considered three objectives applicable to transductive active learning, where the target inputs are known up front: what he called the expected marginal information gain (EMIG), expected joint information gain (EJIG) and expected mean marginal information gain (EMMIG). The first of these is the EIG in the output, y , corresponding to a known target input, x , from observing d :

$$\text{EMIG}(\xi, x) = \text{EIG}_y(\xi, x) \quad (53.34)$$

$$= \text{KL}[p(d, y; \xi, x) \parallel p(d; \xi)p(y; x)]. \quad (53.35)$$

EJIG and EMMIG extend this to a set of $N > 1$ known target inputs. EJIG has the same form as EMIG but replaces (x, y) with $(x_{1:N}, y_{1:N})$. Meanwhile EMMIG is a weighted sum of the EMIG values across $x_{1:N}$, using weights $w_{1:N} \in \mathbb{R}_+^N$:

$$\text{EMMIG}(\xi, x_{1:N}, w_{1:N}) = \sum_{n=1}^N \bar{w}_n \text{EMIG}(\xi, x_n) \quad (53.36)$$

where $\bar{w}_n = w_n / \sum_{n'=1}^N w_{n'}$ is a normalised version of the weight on x_n . MacKay [1107] proposed estimators of these three predictive EIGs for regression with a stochastic neural network; MacKay [1106] added EMIG and EMMIG estimators for classification.

While EJIG was used by for example Guestrin et al. [681], Kirsch et al. [899] and Krause et al. [925], and EMMIG by MacKay [1106] and Wang et al. [1702], they have otherwise been relatively neglected in the active-learning literature. BALD is much more often used to represent Bayesian active learning in empirical investigations.

General predictive EIG Bickford Smith et al. [154] later re-emphasised the idea of maximising EIG in predictions but also importantly extended beyond the transductive setting, allowing for cases where the target inputs are not known ahead of time. Their proposed objective, the expected predictive information gain (EPIG), is the EIG in an unknown target input-output pair, (x, y) , from observing d :

$$\text{EPIG}(\xi) = \text{EIG}_{(x,y)}(\xi) \quad (53.37)$$

$$= \text{KL}[p(x)p(d, y; \xi, x) \parallel p(d; \xi)p(y; x)] \quad (53.38)$$

$$= \mathbb{E}_{p(x)}[\text{EMIG}(\xi, x)]. \quad (53.39)$$

In the general case, $p(x)$ is a belief distribution over unknown target inputs, the construction of which is flexible. This belief distribution can also be specialised to recover EMIG and EMMIG: for the former we define it to be a point mass on the single known target input, x ; for the latter we define it to be a discrete distribution over $x_{1:N}$, with probability masses $\bar{w}_{1:N}$.

Variance-based methods

Whereas maximising a predictive EIG corresponds to a focus on Shannon entropy in target outputs, another popular class of methods focuses on variance in those outputs. A frequentist-style method proposed by Cohn et al. [323] has drawn the most attention in the active-learning literature, but also notable are earlier Bayesian-style methods.

Bayesian-style methods A pair of methods follows the Bayesian approach of aiming for maximal uncertainty reduction or minimal future uncertainty (53.2.2), except that they use a generically updated belief distribution in place of the posteriors used in Bayesian objectives. This is a key distinction between these methods and information-based methods, in addition to the use of variance rather than Shannon entropy.

Cohn [320] proposed maximising a form of expected variance reduction¹¹ (EVR):

$$\text{EVR}(\xi) = \mathbb{E}_{p(x)p(d;\xi)} [\mathbb{V}_{p(y;x)}[y] - \mathbb{V}_{p(y;x,d)}[y]]. \quad (53.40)$$

Noting that $\mathbb{V}_{p(y;x)}[y]$ does not vary with ξ , an equivalent approach is to minimise the form of expected future variance¹² (EFV) proposed by Cohn et al. [322]:

$$\text{EFV}(\xi) = \mathbb{E}_{p(x)p(d;\xi)} [\mathbb{V}_{p(y;x,d)}[y]]. \quad (53.41)$$

Frequentist-style method The alternative form of EFV proposed by Cohn et al. [323] follows the frequentist approach of averaging over existing data as well as future data. In particular it takes any inputs, $x_{d_{<t}}$, whose labels, $y_{d_{<t}}$, are already known (i.e. within $d_{<t} = d_{0:t-1}$) and averages over alternative labels, $y'_{d_{<t}}$, for those inputs, which we refer to as label resampling (LR):

$$\text{EFV-LR}(\xi_t, x_{d_{<t}}, z) = \mathbb{E}_{p(x;z)} \left[\mathbb{V}_{p(d_t; \xi_t, z)p(y'_{d_{<t}}; x_{d_{<t}}, z)} [f(x; (x_{d_{<t}}, y'_{d_{<t}}, d_t))] \right] \quad (53.42)$$

where $f(x; d)$ denotes a deterministic predictive model trained on data d . This objective is derived using a frequentist-risk formulation along with assumptions on data-gathering and model training, hence its explicit dependence on an index over data-generating distributions, z [1370].

Other uncertainty-based methods

All of the methods we have introduced so far depend on some notion of future uncertainty in either a model's parameters or its predictions. Another set of methods is based on *current uncertainty in the data we are acquiring*

¹¹Cohn [320] used a model for which $\mathbb{V}_{p(y;x,d)}[y]$ only depends on the input component of d but not the label component. We present an objective that covers a more general model class.

¹²Both Cohn et al. [322] and Cohn et al. [323] were ambiguous in their definitions of predictive variance, so we present our best-guess interpretation of their mathematical expressions. We also relax their assumption of knowing the true distribution over target inputs, using a model in its place.

as a function of the design, ξ . In entropy sampling [1493] we maximise

$$\text{Entropy}(\xi) = H[p(d; \xi)], \quad (53.43)$$

while in least-confidence sampling [1493] we minimise

$$\text{Confidence}(\xi) = \max_{d \in \mathcal{D}} p(d; \xi) \quad (53.44)$$

and in margin sampling [1467] we minimise

$$\text{Margin}(\xi) = p(d = d^{(1)}; \xi) - p(d = d^{(2)}; \xi) \quad (53.45)$$

where $d^{(1)} = \arg \max_{d' \in \mathcal{D}} p(d'; \xi)$ and $d^{(2)} = \arg \max_{d' \in \mathcal{D} \setminus \{d^{(1)}\}} p(d'; \xi)$.

Earlier use of this general approach includes the work of Lewis and Catlett [1018] and Lewis and Gale [1019], who focused on binary classification (for which these methods are equivalent in their ranking of designs) and on batch acquisition (selecting $B > 1$ inputs at a time). To select a batch, they first used their model to assign binary pseudolabels to the individual candidate inputs, then split the candidate inputs into two sets based on these pseudolabels, and finally selected the $B/2$ highest-uncertainty inputs from each set.

As noted in 53.3.1, these methods are hard to motivate from a decision-theoretic perspective, regardless of whether $p(d; \xi)$ is derived from a Bayesian model or not. Perhaps the closest we can get is to recognise that entropy sampling is equivalent to maximising the EIG in some variable, ψ , if the model's predictions of d given ξ and ψ have equal entropy for all (ξ, ψ) pairs. In this special case we have

$$\text{EIG}_\psi(\xi) = H[p(d; \xi)] - \mathbb{E}_{p(\psi)}[H[p(d|\psi; \xi)]] = \text{Entropy}(\xi) + \text{constant} \quad (53.46)$$

where $p(d; \xi) = \mathbb{E}_{p(\psi)}[p(d|\psi; \xi)]$. Setting aside the strong restriction on the model, there is generally no computational advantage to entropy sampling when it comes to Monte Carlo estimation of the objective: each instance of $p(d|\psi; \xi)$ with some sampled ψ can be used for both terms in this writing of the EIG, so the number of instantiations is the same regardless of whether we use the EIG or the entropy [300].

Error-based methods

Pushing back against methods including query by committee (53.4.2) and least-confidence sampling (53.4.2), Roy and McCallum [1429] advocated focusing on *predictive error*. Each of their two proposed objectives (to be minimised) is a form of expected future error (EFE) in classifying inputs from a given pool, $\mathcal{X}_{\text{pool}}$:

$$\text{EFE}(\xi, \mathcal{X}_{\text{pool}}) = \frac{1}{|\mathcal{X}_{\text{pool}}|} \sum_{x \in \mathcal{X}_{\text{pool}}} \mathbb{E}_{p(d; \xi)}[\ell_y(p(y; x, d), p(y; x, d))] \quad (53.47)$$

where $\ell_y : \mathcal{P}(\mathcal{Y}) \times \mathcal{P}(\mathcal{Y}) \rightarrow \mathbb{R}$ is a loss measuring the mismatch between a predictive distribution and a reference distribution, and $p(y; x, d)$ is a predictive distribution over outputs given x and d (which, as in 53.4.2, is not necessarily produced by a Bayesian update on d). They highlighted two key steps in constructing this objective: the use of an empirical average over $x \in \mathcal{X}_{\text{pool}}$ to approximate an expectation over unknown target inputs; and the use of $p(y; x, d)$ in place of an external reference distribution.

They instantiated this general form of EFE with two different losses. Defining $\ell_y^{\log}(q, p) = \mathbb{E}_{p(y)}[-\log q(y)]$ and substituting this into the general form gives

$$\text{EFE}^{\log}(\xi, \mathcal{X}_{\text{pool}}) = \frac{1}{|\mathcal{X}_{\text{pool}}|} \sum_{x \in \mathcal{X}_{\text{pool}}} \mathbb{E}_{p(d; \xi)}[H[p(y; x, d)]] , \quad (53.48)$$

while using $\ell_y^{0/1}(q, p) = \mathbb{E}_{p(y)}[\mathbb{I}(y \neq \arg \max_{y' \in \mathcal{Y}} q(y'))]$ gives

$$\text{EFE}^{0/1}(\xi, \mathcal{X}_{\text{pool}}) = \frac{1}{|\mathcal{X}_{\text{pool}}|} \sum_{x \in \mathcal{X}_{\text{pool}}} \mathbb{E}_{p(d; \xi)} \left[1 - \max_{y \in \mathcal{Y}} p(y; x, d) \right]. \quad (53.49)$$

Although they did not explicitly aim to target future uncertainty in their approach, both of these objectives are measures of expected uncertainty in $p(y; x, d)$, with uncertainty measured by Shannon entropy in the log-loss case and by a confidence-based measure (53.4.2) in the zero-one-loss case. As with the Bayesian-style variance-based objectives presented in 53.4.2, the use of generic belief updating in the definition of these EFE objectives sets them apart from EIG objectives in the general case.

In later work Zhu et al. [1879] used a transductive variant of the zero-one EFE objective, replacing $\mathcal{X}_{\text{pool}}$ with a separate set of known target inputs. Mussmann et al. [1218] similarly used separate target inputs in their EFE objectives but also deviated from the original EFE objectives in specifying $p(y; x, d) = p(y|d; x) \propto p(d, y; x)$ to be the result of a Bayesian update derived from an explicit joint model.

Coverage-based methods

Recent years have seen considerable interest in a style of approach distinct from those we have discussed so far, namely maximising some notion of *coverage* within a feature space, $\Phi = \{\phi(x) \mid x \in \mathcal{X}\}$, such as that of a deep neural network trained on $d_{0:t-1}$. These tend to be motivated by an aim to reduce frequentist risk, with theoretical results connecting this to coverage under strong assumptions on the setup, such as using a one-nearest-neighbour model on a deterministic classification problem [73, 1815].

Much of the interest in coverage-based methods is due to work by Sener and Savarese [1489]. Their “greedy k centres” method builds up a batch of B inputs, $\xi = x_{1:B}^+ \in \mathcal{X}_{\text{pool}}^B$, through B iterations of selecting the new input, x_b^+ , that maximises the minimal distance from inputs that are already labelled or due to be labelled:

$$\text{MinimalDistance}(x_b^+, \mathcal{X}_{\text{lab}}) = \min_{x \in \mathcal{X}_{\text{lab}}} \|\phi(x_b^+) - \phi(x)\|_2 \quad (53.50)$$

where $\mathcal{X}_{\text{lab}} = \{x \mid x \in (x_{d_{<t}}, x_{<b}^+)\}$ with $x_{d_{<t}}$ denoting inputs in $d_{<t} = d_{0:t-1}$ whose labels are already known, and $x_{<b}^+ = x_{1:b-1}^+$ denoting inputs already in the current batch.

More recent work has proposed other notions of coverage. MaxHerdning [73] considers the similarity between each pool input and its most similar labelled or selected input, and selects the new input, x_b^+ , that maximises the average of those similarities, where the similarity is defined by a real-valued function, such as a kernel, $k : \Phi \times \Phi \rightarrow \mathbb{R}_{\geq 0}$:

$$\text{KernelCoverage}(x_b^+, \mathcal{X}_{\text{lab}}, \mathcal{X}_{\text{pool}}, k) = \frac{1}{|\mathcal{X}_{\text{pool}}|} \sum_{x \in \mathcal{X}_{\text{pool}}} \max_{x_{\text{lab}} \in \mathcal{X}_{\text{lab}} \cup \{x_b^+\}} k(\phi(x), \phi(x_{\text{lab}})). \quad (53.51)$$

The intuitive goal is to construct a labelled dataset that closely matches the pool in feature space, corresponding to a high average similarity between pool inputs and labelled inputs. ProbCover [1815] is a special case of MaxHerdning where the similarity function simply measures whether two inputs are within a specified distance, $\delta \in \mathbb{R}_+$, of each other in feature space, namely $k(\phi_1, \phi_2) = \mathbb{I}(\|\phi_1 - \phi_2\|_2 \leq \delta)$.

Another related method is TypiClust [698], which is motivated by the idea of seeking inputs that are diverse but also representative of the pool. It uses clustering to promote diversity, and a maximisation of a typicality score to promote representativeness. More specifically it splits $\mathcal{X}_{\text{pool}}$ into clusters then performs B iterations of selecting a cluster (minimising the number of labelled inputs; splitting ties by maximising cluster size) and within the cluster, $\mathcal{X}_{\text{clust}}$, selecting the input, x_b^+ , that maximises “typicality” within a neighbourhood of size $1 \leq k \leq |\mathcal{X}_{\text{clust}}|$:

$$\text{Typicality}(x_b^+, \mathcal{X}_{\text{clust}}, k) = \frac{k}{\sum_{x \in k\text{-NN}_{\phi}(x_b^+, \mathcal{X}_{\text{clust}})} \|\phi(x_b^+) - \phi(x)\|_2} \quad (53.52)$$

where $k\text{-NN}_{\phi}(x_b^+, \mathcal{X}_{\text{clust}})$ denotes the k nearest neighbours to x_b^+ within $\mathcal{X}_{\text{clust}}$, which are found by sorting $x \in \mathcal{X}_{\text{clust}}$ by $\|\phi(x_b^+) - \phi(x)\|_2$.

Gradient-based methods

Also notable for their recent popularity are methods based on *gradients* of losses with respect to model parameters, which have been put forward with a range of motivations. Settles et al. [1495] proposed seeking inputs that are expected to most change the parameters, ψ , of a model as the result of a gradient-based update on the labelled input. They defined the expected gradient length (EGL) for a candidate input, x^+ , as

$$\text{EGL}(x^+) = \mathbb{E}_{p(y^+; x^+, \psi)} \left[\left\| \text{vec} \left(\frac{\partial}{\partial \psi} \ell_y(p_{y^+}(\cdot; x^+, \psi), y^+) \right) \right\|_2 \right] \quad (53.53)$$

where $\text{vec}(\cdot)$ denotes vectorisation (i.e. flattening) and $\ell_y : \mathcal{P}(\mathcal{Y}) \times \mathcal{Y} \rightarrow \mathbb{R}$ is a loss function on probabilistic predictions. They suggested maximising this objective to select a single input, but did not propose a scheme for selecting a batch of $B > 1$ inputs at once.

Ash et al. [54] later paid particular attention to batch acquisition for neural networks, seeking a way to select multiple high-uncertainty inputs (as in 53.4.2) that satisfy some notion of diversity. Similar to Settles et al. [1495], they used gradients of a loss function evaluated with model-generated pseudolabels, with the idea of capturing potential model change. But their desire for diversity led them to use these gradients in a different way, namely forming so-called gradient embeddings of inputs and then passing these to k -means++ [50], an algorithm designed for choosing the initial cluster centres in k -means clustering. More specifically their method, BADGE (“batch

active learning by diverse gradient embeddings”), selects a batch of B inputs by running k -means++ with $k = B$ on a set of embeddings, $\mathcal{E}$, defined as

$$\mathcal{E} = \left\{ \text{vec} \left(-\frac{\partial}{\partial \psi'_L} \log p(y = y^{(1)}; x, \psi) \right) \mid x \in \mathcal{X}_{\text{pool}}, x \notin x_{d_{<t}} \right\} \quad (53.54)$$

where $\psi = \psi'_{1:L}$ parameterises an L -layer neural network, $y^{(1)} = \arg \max_{y \in \mathcal{Y}} p(y; x, \psi)$ is the output assigned highest weight by the neural network for input x , and $x_{d_{<t}}$ are the inputs in $d_{<t} = d_{0:t-1}$ whose labels are already known.

Ash et al. [52] later drew a connection between the embeddings used in BADGE and the Fisher information matrix of the neural network at x ,

$$F(x, \psi) = \mathbb{E}_{p(y; x, \psi)} \left[\left(\frac{\partial}{\partial \psi} \log p(y; x, \psi) \right) \left(\frac{\partial}{\partial \psi} \log p(y; x, \psi) \right)^\top \right], \quad (53.55)$$

and argued that a more principled approach is to minimise an objective sometimes known as the Fisher-information ratio (FIR; [1493, 1494]):

$$\text{FIR}(\xi, x_{1:N}, \psi) = \text{Tr}(F(x_{1:N}, \psi)(F(x_{d_{<t}}, \psi) + F(\xi, \psi))^{-1}) \quad (53.56)$$

where $x_{1:N}$ is a known set of target inputs and we note that $F(x'_{1:N}, \psi) = \sum_{n=1}^N F(x'_n, \psi)$. The use of known target inputs makes this a transductive objective. A non-transductive form, replacing $F(x_{1:N}, \psi)$ with an expectation of $F(x, \psi)$ with respect to an unknown target input, x , was proposed by Fukumizu and Watanabe [562]. Both forms of the objective have been studied elsewhere in the literature [563, 767, 1468, 1493, 1859], with some work highlighting a connection between FIR and a form of frequentist risk.

53.4.3 Methodological considerations

Finally, having covered a range of existing active-learning methods, we now highlight some important issues relating to their use and to the development of new methods.

Targeting predictions A key factor of variation across the methods in 53.4.2 is how directly they target the predictions of interest. One notable case is BALD, which targets uncertainty in a set of stochastic model parameters (53.4.2), not uncertainty in the model’s predictions directly, and can thus be of limited use in improving predictive performance [154, 155, 559, 1107]. In other cases the predictive focus is present in the high-level motivation for the method, but the concrete method is not directly derived from a mathematical formulation of that motivation (e.g. coverage-based methods). This is an important consideration when assessing existing methods and designing new ones. A clear connection to the decision problem of interest is a necessary condition for a method to be principled, although it is also not sufficient for good performance in practice.

Belief updating In 53.3.1 we highlighted that principled uncertainty-based objectives are critically expressed in terms of *future* beliefs after updating on new data; and in 53.4.2 we noted that EIGs assume those beliefs are derived from the joint model underlying the EIG, while variance-based and error-based objectives assume the beliefs are produced by a generic procedure (e.g. re-running a training algorithm). Interestingly, while the Bayesian-update assumption is theoretically principled (and can be applied to cases beyond targeting Shannon entropy), the way beliefs are actually updated in practical active learning warrants attention to the assumption. For example, it is standard practice in information-based active learning to produce final predictions after data-gathering using a generic predictive algorithm [154, 155, 569, 784, 785, 898, 900, 1106, 1107]. There is thus a mismatch between the belief update assumed during data-gathering and the belief update used after data-gathering. At the same time, information-based methods often perform well in empirical investigations, and currently the literature lacks empirical evidence on the influence the mismatch has on performance. This warrants further investigation.

Constructing accurate models Many of the methods we covered in 53.4.2 are based on probabilistic models: models of the target inputs we will make predictions on; models of the future data we will gather; and, in frequentist approaches, models of the data we already have. All of these models are subjective and in the general case will not match reality [156]. The extent of that mismatch is a critical determinant of how well the methods perform

in practice. Using more accurate models of course improves baseline performance, but it can also make data-gathering methods more effective. For example, switching from a fully supervised to a semi-supervised model that incorporates unlabelled data (53.4.1) has been shown to significantly improve the effectiveness of information-based methods [155]. Whether it is through semi-supervised learning or not, any approach that allows us to construct better models has the potential to benefit data-gathering. Designing predictive algorithms that work well in low-data regimes is thus a research priority.

An important factor in designing such predictive algorithms is that elements of an actively acquired data sequence are not independent and identically distributed, leading to statistical bias in models produced by standard empirical-risk minimisation [147, 304, 373, 571, 1107, 1343]. Reweighted risk estimators can be used to prevent this bias in the context of pool-based active learning, but interestingly these do not always lead to better predictions, with the bias actually helping in some cases [506].

53.5 Conclusion

We opened this chapter by highlighting the enormous value that could be unlocked by improving data-gathering methods. To that end, we have presented context and conceptual tools that we hope will accelerate progress. Our approach has been to emphasise the framework of decision theory, which enables us to directly target what we care about, and which fundamentally motivates the use of probabilistic methods. Recognising the primacy of decision-making—and moving beyond a narrow focus on inference, which cannot be decoupled from decision-making in anything but the simplest of problems [956]—is crucial if the field of Bayesian deep learning is to achieve the practical relevance and mainstream impact it aspires to [1292].

In using decision theory to explore notions of optimality in data-gathering, we have demonstrated deep connections between loss functions (or corresponding utility functions), proper scoring rules, and generalised forms of entropy and information gain. We have also drawn attention to a crucial insight: while the popular approach of minimising expected posterior uncertainty is theoretically optimal, so too is an approach that avoids posteriors entirely and that also naturally targets downstream decision-making [1424]. Thus we have seen the value of a foundational perspective to make sense of a wide array of ideas but also to directly influence the design of practical methods.

Challenges remain in developing methods ready for widespread deployment. One of these is dealing with *model misspecification*, in which no configuration of the model we are using can recover the true data-generating process. While this has received some consideration with respect to experimental design [2, 71, 98, 227, 513, 546, 641, 1280, 1572, 1590], it remains drastically understudied given the prevalence and impact of model misspecification in practice.

A second challenge, solving which will rely on addressing model misspecification, is scaling up from simple proof-of-concept demonstrations to complex problems, such as medical decision-making or supporting creative work, where the spaces of designs and data are high-dimensional and especially hard to map between. This requires innovation to combine two threads of work: that which has demonstrated more sophisticated methods [552] and that which has tackled harder problems [300].

One complex problem that warrants special attention is *active testing* [922, 923, 1464], in which the goal is to evaluate the behaviour of a model, conventionally in terms of its predictive performance. While progress has been made in scaling up active testing to large models [132] and providing confidence intervals on performance estimates [56], there is more work to be done to handle more complex notions of model behaviour than the basic measures of predictive performance that have been the focus of most prior work.

Another problem that we have not directly covered in this chapter is *Bayesian optimisation* [584], in which the goal is to produce an action that optimises an unknown objective function. Bayesian optimisation is like active learning in being a particular form of experimental design around which an effectively separate community and literature has grown; in this case its distinctive features include widespread use of Gaussian-process models. There are thus parallel threads of work and inevitably cases where one field's solutions could be useful in another. Efforts to identify such cases and translate ideas between fields could be fruitful.

There is therefore a considerable amount of work to be done. At the same time, we find ourselves increasingly well equipped to make progress. Ever-improving algorithms [450, 1212, 1213, 1632] let us construct more capable models, and ever-cheaper computation [340, 494, 1249] lets us simulate at greater scale. Poorly designed methods will squander the dividends these trends yield, so we cannot stick with what we have and expect our problems to be solved. If instead we pursue the right ideas, a great reward is within reach.

Chapter 54

Causal-aware BDL

54.1 Introduction

Bayesian deep learning (BDL) equips flexible neural models with a coherent probabilistic treatment of parameters, latent variables, and predictions, as discussed in previous chapters. Combined with deep neural networks, BDL has achieved remarkable success in recognising and generating images, learning representations, and translating languages by exploiting statistical correlations in data. However, many such models remain primarily associational: they learn what tends to *co-occur*, rather than what *produces* what. This distinction becomes critical when the environment changes, when a model is used to support decisions, or when we ask hypothetical questions about interventions. Correlations describe what is observed; causality aims to describe what would happen under changes to the system.

Causality therefore focuses on the mechanisms that generate data. It asks not only whether two variables move together, but whether one *causes* the other and how that influence would change under different interventions. For example, does a new drug improve recovery, or are healthier patients simply more likely to receive it? In practice, causal analysis involves two intertwined tasks. The first is *causal discovery*: learning the underlying cause–effect structure among variables from the data or active experimentation. The second is *causal inference*: estimating the effect of an intervention once a causal structure is known, hypothesised, or partially identified. In essence, causal discovery asks “who causes whom”, while causal inference asks “by how much” and “what would happen if we intervened”.

To make these questions mathematically precise, causal reasoning relies on *structural equation models* (SEMs), sometimes called *structural causal models* (SCMs). An SEM describes how each variable is generated as a function of its direct causes together with an independent source of noise. This formulation encodes the idea that a system is composed of modular mechanisms: interventions modify one mechanism while leaving the others unchanged. SEMs therefore provide a bridge between observed data and hypothetical reasoning, making interventions and counterfactuals mathematically coherent.

Causal graphs provide a complementary language for representing the cause–effect structure. A causal graph consists of nodes, representing variables, and directed edges, representing direct causal relationships. Although cyclic relations can arise in certain dynamical systems, this chapter primarily focuses on *directed acyclic graphs* (DAGs), which are the standard setting for the methods considered here. Graphs also provide a powerful language for inference: they encode conditional independencies, reveal possible confounding pathways, and help determine when causal effects can be identified from observational data.

Seen from this perspective, causality is not an alternative to probabilistic modelling, but a natural extension of it. In this view, Bayesian methods become particularly useful. They offer principled tools for quantifying uncertainty over parameters, causal structures, and variables, while causal modelling provides the language for reasoning about interventions and changes in the underlying system. Combining the two yields a framework for reasoning under uncertainty about causal mechanisms themselves: we can express prior knowledge about possible causal structures, update it with data, and propagate uncertainty into downstream causal estimates.

In this chapter, we illustrate how the tools from Bayesian deep learning can be combined with causal reasoning for both static and time-series data. We first introduce the basic causal language used throughout the chapter, including causal graphs, SEMs, causal estimands, and identifiability. We then describe two representative methods for static data: *DECI* and *BayesDAG*. Next, we move to the temporal domain, introducing *Rhino* for discrete-time data and *SCOTCH* for continuous-time data. Finally, we discuss fixed-point causal models and foundation-model approaches to causal inference.

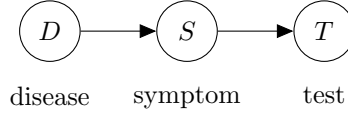


Figure 54.1: A DAG for $D \rightarrow S \rightarrow T$. The Markov property implies $D \perp T \mid S$.

54.2 Basics for causal machine learning

54.2.1 Basics and notations

Causal modelling seeks to understand and reason about how changes in one part of a system influence another. Causal analysis goes further, asking what would happen to a variable Y if we were to *intervene* on another variable X .

This distinction between observation and intervention lies at the heart of causality [1307, 1318].

Three levels of questions. Following Pearl [1307], it is helpful to distinguish three kinds of questions:

1. *Associational*: What relationships are present in the data, e.g. $P(Y \mid X)$?
2. *Interventional*: What would happen under an external manipulation, e.g. $P(Y \mid \text{do}(X=x))$?
3. *Counterfactual*: What would have happened had something been different for a specific instance?

The first can be answered from observational data alone; the latter two require additional causal assumptions, often represented by a graph.

Variables and notation. Here, we focus primarily on the static-data regime; most concepts extend naturally to temporal settings. Let $\mathbf{X} = (X_1, \dots, X_D)$ denote D random variables of interest, jointly distributed according to $p(\mathbf{X})$. We write lowercase letters x_i for realisations of X_i . A bold symbol $\mathbf{x}$ denotes a vector of all variables, and for a subset $\mathcal{S} \subseteq \{1, \dots, D\}$ we use $X_{\mathcal{S}} = \{X_i : i \in \mathcal{S}\}$ and $x_{\mathcal{S}}$ analogously.

Graphs as causal structure. A *directed graph* $\mathbf{G} = (V, E)$ with node set $V = \{1, \dots, D\}$ and directed edges $E \subseteq V \times V$ is used to encode causal relations between variables. An edge $i \rightarrow j$ means that X_i is a *cause* of X_j . For each node i , the *parents* and *children* sets are

$$\mathbf{Pa}(i) = \{j \in V : (j, i) \in E\}, \quad \mathbf{Ch}(i) = \{j \in V : (i, j) \in E\}.$$

Thus, the parents of node i are precisely the nodes j with edges $j \rightarrow i$. We denote the *ancestors* $\mathbf{An}(i)$ as all nodes from which a directed path leads to i , and the *descendants* $\mathbf{De}(i)$ as all nodes reachable from i via directed paths. When the graph has no directed cycles (i.e. no node is its own ancestor), it is called a *directed acyclic graph* (DAG). Throughout this chapter, we assume $\mathbf{G}$ is a DAG. An adjacency matrix encodes the causal graph: for a binary square matrix $\mathbf{G}$, $\mathbf{G}_{ij} = 1$ denotes $i \rightarrow j$, whereas $\mathbf{G}_{ij} = 0$ denotes the absence of that edge. In some cases, $\mathbf{G}$ can also be real-valued, representing the weight of the associated directed edge.

Paths and separation. A *path* in $\mathbf{G}$ is a sequence of distinct nodes connected by edges, regardless of their orientation. A triple of nodes (A, B, C) can form a *chain* $A \rightarrow B \rightarrow C$, a *fork* $A \leftarrow B \rightarrow C$, or a *collider* $A \rightarrow B \leftarrow C$, where B is the collider. Conditioning on a non-collider blocks a path, whereas conditioning on a collider or one of its descendants can open it. The concept of *d-separation* formalises when a set of variables Z blocks every path between X and Y . We write $X \perp_{\mathbf{G}} Y \mid Z$ for this graphical relation; under the causal Markov property, it implies the corresponding conditional independence in the induced distribution.

From graphs to probability. Graphs alone are qualitative; to reason quantitatively, we link them to probability distributions that respect the graph's structure. Roughly, each node depends only on its parents, and the joint distribution factorises accordingly:

$$p(x_1, \dots, x_D) = \prod_{i=1}^D p(x_i \mid x_{\mathbf{Pa}(i)}).$$

This factorisation will later arise naturally from the *structural equation models* (SEMs) introduced in the next section and is formalised by the *causal Markov property*. The key takeaway is that a DAG represents how probabilistic dependencies arise from an underlying system of cause–effect mechanisms.

Running example. Figure 54.1 shows the running example $D \rightarrow S \rightarrow T$. Under faithfulness, D and T are marginally dependent; the Markov property implies that they become conditionally independent given S , because conditioning on S blocks the path between them.

This graphical language provides the foundation for all subsequent sections: structural equations will make the above factorisation explicit, interventions will modify parts of it, and causal discovery will aim to recover $\mathbf{G}$ from data.

54.2.2 Structural Equation Model

In probabilistic modelling, we often describe data generation by specifying priors and likelihoods for latent variables. A *structural equation model* (SEM) plays an analogous role for causality, but its primitives are explicit *cause–effect mechanisms*. Each variable is expressed as a deterministic function of its direct causes, together with an exogenous noise term that accounts for unexplained variation. The collection of these mechanisms jointly defines how the system generates observations.

Definition 54.1 (Structural Equation Model). *Let $\mathbf{X} = (X_1, \dots, X_D)$ denote a set of random variables. A structural equation model consists of a family of measurable functions $\{f_i\}_{i=1}^D$ and jointly independent noise variables $\{U_i\}_{i=1}^D$ such that*

$$X_i := f_i(X_{\mathbf{Pa}(i)}, U_i), \quad i = 1, \dots, D, \quad (54.1)$$

where $\mathbf{Pa}(i) \subseteq \{1, \dots, D\} \setminus \{i\}$ denotes the parents (direct causes) of X_i . The directed edges $j \rightarrow i$ induced by the parent sets form a graph $\mathbf{G}$ on nodes $\{1, \dots, D\}$, called the causal graph. When $\mathbf{G}$ contains no directed cycles, it is a directed acyclic graph (DAG).

The exogenous variables U_i represent influences external to the system and are often referred to as *noise terms*. Given mutually independent $\{U_i\}$, Equation 54.1 defines a joint distribution over all variables. Restricting the causal graph to be a DAG offers many advantages that facilitate analysis. However, real-world systems often contain feedback: demand $\rightleftharpoons$ price, predator $\rightleftharpoons$ prey. A *cyclic* SEM allows directed cycles in $\mathbf{G}$. This can introduce challenges; for example, the system may have no solution or multiple equilibria without additional assumptions. Therefore, this chapter focuses primarily on the DAG setting. When the causal graph is a DAG, sampling proceeds recursively in topological order: one first samples the root nodes (those without parents) and then successively applies each mechanism f_i using already generated parent values.

This construction induces a joint density of the form

$$p(x_1, \dots, x_D) = \prod_{i=1}^D p(x_i \mid x_{\mathbf{Pa}(i)}), \quad (54.2)$$

where $p(x_i \mid x_{\mathbf{Pa}(i)})$ is the distribution obtained by pushing forward the noise U_i through $f_i(\cdot, \cdot)$. Equation (54.2) expresses that, in an SEM, each variable depends probabilistically only on its parents. This factorisation follows from the *causal Markov property*.

Definition 54.2 (Causal Markov Property). *Given a DAG $\mathbf{G}$, each node X_i is independent of its non-descendants conditional on its parents:*

$$X_i \perp \text{ND}(i) \mid X_{\mathbf{Pa}(i)}.$$

Here, $\text{ND}(i)$ denotes the non-descendants of node i : all nodes that are not reachable from i via a directed path in $\mathbf{G}$. This assumption encodes that parents *cut off* other upstream information about a node.

Two other assumptions required in this chapter are *causal sufficiency* and *causal minimality and faithfulness*.

Definition 54.3 (Causal Sufficiency). *There are no unobserved common causes of any two or more variables in the variable set $\{X_i\}_{i=1}^D$. Equivalently, there are no latent confounders in the causal graph $\mathbf{G}$.*

This is an important assumption that rules out the possibility of hidden common causes in the system.

Definition 54.4 (Causal Minimality and Faithfulness). *For a distribution $p(\mathbf{X})$ induced by an SEM with causal graph $\mathbf{G}$, we say the model satisfies causal minimality if $p(\mathbf{X})$ is Markov with respect to $\mathbf{G}$ but is not Markov with respect to any proper subgraph of $\mathbf{G}$. Faithfulness asserts that all and only conditional independencies in $p(\mathbf{X})$ arise via d -separation in $\mathbf{G}$.*

Intuitively, minimality requires that no edge in $\mathbf{G}$ is superfluous for representing the distribution, while faithfulness rules out conditional independencies caused only by special parameter cancellations. Together, these assumptions allow us to read statistical independencies from the graph while excluding degenerate coincidences [1318].

54.2.3 Structural identifiability

An essential question in causal discovery is whether the causal structure underlying a set of variables can be recovered from data. When we observe samples from the joint distribution of $\mathbf{X} = (X_1, \dots, X_D)$, there may exist several distinct structural equation models that induce exactly the same distribution. In such cases, it is impossible to determine which causal graph generated the data, even with infinite samples. The concept of *structural identifiability* formalises the conditions under which this ambiguity disappears. Here, the object to be identified is the causal graph from the observational distribution, rather than a particular parameterisation of its mechanisms. This differs from the neural-network parameter non-identifiability caused by symmetries discussed in Chapter 32.

Let $\mathcal{M}$ be a model family of structural equation models as defined in Section 54.2.2, parameterised by functional relations and noise distributions. Each element $(\mathbf{G}, \theta) \in \mathcal{M}$, consisting of a causal graph $\mathbf{G}$ and structural parameters θ , induces an observational distribution $P_{\mathbf{G}, \theta}(\mathbf{X})$. Fix a data-generating model $(\mathbf{G}^*, \theta^*) \in \mathcal{M}$. We say that its causal structure $\mathbf{G}^*$ is *structurally identifiable* within $\mathcal{M}$ if, for every $(\mathbf{G}, \theta) \in \mathcal{M}$,

$$P_{\mathbf{G}, \theta}(\mathbf{X}) = P_{\mathbf{G}^*, \theta^*}(\mathbf{X}) \implies \mathbf{G} = \mathbf{G}^*.$$

Thus, within the assumed model family, even infinite observational data identify the graph but need not identify a unique value of θ .

In general, however, the mapping from structural models to observational distributions is not injective [638]. Different graphs can lead to identical statistical dependencies among variables. Under the causal Markov and faithfulness assumptions, the observational distribution $P(\mathbf{X})$ determines only a *Markov equivalence class* of DAGs rather than a single one. Two DAGs are Markov equivalent if they encode the same set of conditional independences, meaning that every distribution that factorises with respect to one also factorises with respect to the other. To make an SEM structurally identifiable, one common approach is to restrict its flexibility.

Additive noise model The definition of an SEM in Section 54.2.2 is fully general: the noise U_i can interact with parents in arbitrary, even non-additive, ways. This expressiveness comes with costs:

1. *Non-identifiability*: multiple distinct causal graphs and functions can induce the same observational distribution; without structure, recovering the causal graph from observations alone is impossible in general [1318].
2. *Ambiguous noise–signal separation*: without restrictions on how U_i enters, the observational distribution need not determine a unique split between the mechanism and its noise.

For example, any continuous bivariate distribution can be represented in either direction as $Y = F_{Y|X}^{-1}(U | X)$ or $X = F_{X|Y}^{-1}(V | Y)$, where the corresponding $U, V \sim \text{Unif}(0, 1)$ are independent of the conditioning variable. Unrestricted function–noise pairs therefore do not identify the causal direction. *Additive noise models* (ANMs), on the other hand, impose the simplifying form

$$X_i = f_i(X_{\mathbf{Pa}(i)}) + U_i, \quad U_i \perp X_{\mathbf{Pa}(i)}, \quad U_i \perp \{U_k : k \neq i\}, \quad (54.3)$$

ANMs simplify the SEM in a way that can establish identifiability and facilitate estimation of causal quantities under suitable conditions [786].

54.2.4 Causal inference through SEMs

Once a structural equation model has been specified, it provides more than a description of observed dependencies. Because the model explicitly encodes how each variable is generated from its direct causes, it allows us to reason about the consequences of interventions—what would happen if we were to change a variable by external manipulation rather than observation. This ability to predict the effect of hypothetical actions is what distinguishes causal models from purely statistical ones.

In an SEM defined by

$$X_i := f_i(X_{\mathbf{Pa}(i)}, U_i), \quad i = 1, \dots, D,$$

each function f_i represents a stable causal mechanism: it describes how X_i would respond to changes in its parents. Intervening on a variable, for instance setting X_k to a fixed value a , means that the normal mechanism generating X_k is replaced by a constant function. This operation is formalised through the *do-operator* $\text{do}(X_k=a)$, and it defines a new system of equations

$$X_k := a, \quad X_i := f_i(X_{\mathbf{Pa}(i)}, U_i) \text{ for } i \neq k.$$

The corresponding post-intervention distribution is denoted by $P(\mathbf{X} \mid \text{do}(X_k=a))$. Conceptually, this distribution describes what would be observed if X_k were fixed externally, disconnecting it from its usual causes. In the causal graph, this amounts to deleting all incoming edges into node k while leaving the rest of the graph unchanged. Under this intervention, the factorisation of the joint distribution changes accordingly:

$$p(x_1, \dots, x_D \mid \text{do}(X_k=a)) = \delta_a(x_k) \prod_{i \neq k} p(x_i \mid x_{\mathbf{Pa}(i)}), \quad (54.4)$$

where $\delta_a(\cdot)$ is a point mass at the intervened value a . This expression illustrates that the effect of an intervention can be computed directly from the SEM by modifying only the local mechanism of the intervened variable, while all other structural relations remain intact. The notion of a *causal effect* arises naturally from this intervention semantics. For two variables X_i and X_j , the causal effect of setting X_i to x_i on the outcome X_j is defined as the expected value of X_j under that intervention:

$$\mathbb{E}[X_j \mid \text{do}(X_i=x_i)].$$

This expectation can be compared across different intervention values of X_i to quantify the strength or direction of the causal influence.

A particularly useful summary measure is the *average treatment effect* (ATE), which captures the expected change in the outcome when comparing two intervention levels x_i and x'_i of X_i :

$$\text{ATE}(x_i, x'_i) = \mathbb{E}[X_j \mid \text{do}(X_i=x_i)] - \mathbb{E}[X_j \mid \text{do}(X_i=x'_i)]. \quad (54.5)$$

This quantity represents the population-level causal impact of changing X_i from x'_i to x_i . In many applications, however, one is interested in how this effect varies across subpopulations characterised by other variables Z . The corresponding conditional quantity,

$$\text{CATE}(x_i, x'_i; z) = \mathbb{E}[X_j \mid \text{do}(X_i=x_i), Z=z] - \mathbb{E}[X_j \mid \text{do}(X_i=x'_i), Z=z], \quad (54.6)$$

is called the *conditional average treatment effect* (CATE). It measures the causal effect of X_i on X_j for individuals or groups sharing the same value of Z . When the full SEM is known, causal effects such as these can be computed directly by propagating the modified structural equations as in Equation 54.4.

54.2.5 DAG constraints for causal discovery

The previous sections have described how an SEM encodes causal relations, how it supports inference about interventions, and how the ANM family can be structurally identifiable. In practice, however, the causal graph $\mathbf{G}$ itself is rarely known and must be learned from data. Causal discovery, therefore, aims to infer the graph that best explains the observed dependencies among variables. Formally, this is a constrained optimisation problem: among all possible adjacency matrices, only those corresponding to DAGs are admissible. Because the number of possible DAGs grows super-exponentially with the number of variables and acyclicity is a combinatorial constraint, one approach to sidestep this problem is a continuous relaxation or reparameterisation of the DAG constraint.

Continuous relaxation of DAG constraints

Recent developments in differentiable causal discovery reformulate this problem as continuous optimisation with a differentiable DAG constraint, allowing gradient-based methods to scale to higher dimensions.

Let $\mathbf{G} \in \mathbb{R}^{D \times D}$ be a weighted adjacency matrix for D variables. To characterise whether $\mathbf{G}$ is acyclic, NOTEARS [1868] introduces the smooth and differentiable DAG constraint

$$h(\mathbf{G}) = \text{tr}(\exp(\mathbf{G} \odot \mathbf{G})) - D, \quad (54.7)$$

where $\odot$ denotes elementwise multiplication and $\exp$ is the matrix exponential. This constraint computes the total weight associated with closed walks of any length encoded by $\mathbf{G}$. Therefore, $h(\mathbf{G}) \geq 0$, and $h(\mathbf{G}) = 0$ if and only if there is no directed cycle in $\mathbf{G}$. Hence, a standard way to impose DAGness is to augment the learning objective with $h(\mathbf{G})$, translating the constrained optimisation into an unconstrained one.

In practice, this constraint is often used with an augmented Lagrangian to encourage the resulting graph to be acyclic. Suppose $\mathcal{L}_{\text{fit}}(\theta, \mathbf{G})$ is the data-fit objective for a chosen SEM class (e.g., the log-likelihood or ELBO introduced later) with model parameters θ . We wish to solve

$$\max_{\theta, \mathbf{G}} \mathcal{L}_{\text{fit}}(\theta, \mathbf{G}) \quad \text{s.t.} \quad h(\mathbf{G}) = 0. \quad (54.8)$$

The *augmented Lagrangian* replaces the hard constraint with the practical objective

$$\mathcal{L}_{\text{AL}}(\theta, \mathbf{G}; \alpha, \rho) = \mathcal{L}_{\text{fit}}(\theta, \mathbf{G}) - \alpha h(\mathbf{G}) - \frac{\rho}{2} h(\mathbf{G})^2, \quad (54.9)$$

where $\alpha \geq 0$ is the Lagrange multiplier and $\rho > 0$ is the quadratic-penalty coefficient. Optimisation alternates between maximising this objective and updating α , while increasing ρ only when $h(\mathbf{G})$ does not decrease sufficiently. This avoids fixing an excessively large penalty in advance while driving $h(\mathbf{G})$ towards zero.

Reparameterisation of DAGs through node potentials

An alternative strategy is to *reparameterise* the hypothesis space so that every member already encodes a DAG. One idea is to assign a scalar *potential* $p_i \in \mathbb{R}$ to each node i and orient edges *downhill* in that potential order. Formally, define the matrix

$$(\text{grad } \mathbf{p})(i, j) = p_i - p_j, \quad \mathbf{G}(\mathbf{p}) = \text{ReLU}(\text{grad } \mathbf{p}),$$

where ReLU is applied elementwise. Thus, $\mathbf{G}(\mathbf{p})_{ij} > 0$ only if $p_i > p_j$, and $\mathbf{G}(\mathbf{p})$ must define a DAG. Let $\mathbf{W} \in \mathbb{R}^{D \times D}$ be a *skew-symmetric* score matrix ($\mathbf{W} = -\mathbf{W}^\top$) that carries edge strengths. The mapping

$$\gamma(\mathbf{W}, \mathbf{p}) = \mathbf{W} \odot \mathbf{G}(\mathbf{p}) \quad (54.10)$$

yields a *continuously weighted DAG*: every nonzero entry of $\gamma(\mathbf{W}, \mathbf{p})$ points from a higher potential to a lower one, so potentials strictly decrease along any directed path; a cycle would force a net decrease around a loop, which is impossible. Hence, $\gamma(\mathbf{W}, \mathbf{p})$ is acyclic by construction.

For any pair of nodes i, j , at least one of $\mathbf{G}(\mathbf{p})_{ij}$ and $\mathbf{G}(\mathbf{p})_{ji}$ is zero. A skew-symmetric $\mathbf{W}$ therefore reduces the number of free parameters to $\frac{D(D-1)}{2}$. Yu et al. [1826] proved that, with skew-symmetric $\mathbf{W}$, $\gamma(\mathbf{W}, \mathbf{p})$ characterises all possible DAGs. This node-potential formulation is therefore complete and translates the DAG-learning problem into inference over the continuous-valued node potentials $\mathbf{p}$.

54.3 Bayesian Deep Learning for Causality with Static Data

The previous sections have established the theoretical foundations of causality: how structural equation models (SEMs) encode causal mechanisms, how interventions define causal effects, and how directed acyclic graphs (DAGs) constrain possible relations among variables. While these concepts provide the language for causal reasoning, modern data analysis often requires learning both the causal structure and the corresponding functional mechanisms directly from large, high-dimensional observational datasets. This section focuses on static data. Recent advances bridge causal discovery and causal inference by treating both as components of a single generative model represented by an SEM. Traditional causal discovery methods, such as constraint-based or score-based approaches [1318, 1546], typically learn DAGs from conditional-independence tests or scoring functions but do not estimate the functional mechanisms between variables. Recent probabilistic frameworks bring these perspectives together by formulating causal learning as joint inference over the graph and the structural equations. This viewpoint has given rise to differentiable structure-learning methods such as NOTEARS [1868], GraN-DAG [955], and MissDAG [572], as well as Bayesian approaches such as DiBS [1077], BayesDAG [35], and the flow-based *Deep End-to-End Causal Inference* (DECI) framework [590].

In this chapter, we focus on two representative examples of this emerging paradigm: *DECI* and *BayesDAG*. Both operate on static datasets and infer the underlying SEMs with structural identifiability guarantees. DECI provides a unified, end-to-end framework that combines a nonlinear ANM with Bayesian variational inference to jointly learn a graph posterior, structural functions, and exogenous noise distributions. BayesDAG, in contrast, uses stochastic-gradient MCMC (SG-MCMC) and continuous relaxations for sampling-based posterior inference over DAGs.

54.3.1 Deep End-to-End Causal Inference

DECI combines Bayesian variational inference with causal machine learning to represent uncertainty in inferred causal graphs. It applies the variational-inference framework introduced in Chapter 12 to learn a distribution over causal graphs from observational data, infer the associated SEM, and estimate causal quantities. We first describe DECI's Bayesian formulation of an ANM and how it is learned from observational data. We then state the conditions under which DECI recovers the true causal graph with infinite data and explain how the learned model can simulate interventional samples for treatment-effect estimation. The code is available at <https://github.com/microsoft/causica>.

Combining variational inference with ANM

Let $\mathbf{X} = (X_1, \dots, X_D)$ denote the variables, let $\mathbf{Z} = (Z_1, \dots, Z_D)$ denote mutually independent exogenous noise, and let $\mathbf{G} \in \{0, 1\}^{D \times D}$ encode a DAG using the convention $\mathbf{G}_{ij} = 1$ for $i \rightarrow j$. Let θ collect the parameters of the structural mechanisms f_i and the exogenous noise model. DECI assumes the ANM introduced in Equation 54.3:

$$X_i = f_i(X_{\mathbf{Pa}(i; \mathbf{G})}; \theta) + Z_i. \quad (54.11)$$

We rewrite it as the invertible map $\mathbf{Z} = \mathbf{g}_{\mathbf{G}}(\mathbf{X}; \theta) := \mathbf{X} - \mathbf{f}_{\mathbf{G}}(\mathbf{X}; \theta)$. In a topological ordering of the DAG, each f_i depends only on variables preceding X_i . The Jacobian $\partial \mathbf{g}_{\mathbf{G}} / \partial \mathbf{X}$ is therefore triangular with unit diagonal, so its determinant is one [590]. The likelihood consequently factorises as

$$p_{\theta}(\mathbf{X} \mid \mathbf{G}) = p_{\mathbf{Z}}(\mathbf{g}_{\mathbf{G}}(\mathbf{X}; \theta)) = \prod_{i=1}^D p_{Z_i}(g_{\mathbf{G}, i}(\mathbf{X}; \theta)), \quad (54.12)$$

where the first equality follows from the change-of-variables formula and $g_{\mathbf{G}, i}$ is the i th component of $\mathbf{g}_{\mathbf{G}}$. The function $\mathbf{f}_{\mathbf{G}} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ must respect the adjacency relations in $\mathbf{G}$: if $\mathbf{G}_{ji} = 0$, then f_i must satisfy $\partial f_i(\mathbf{X}) / \partial X_j = 0$. DECI uses the flexible parameterisation

$$f_i(\mathbf{X}) = \zeta_i\left(\sum_{j=1}^D \mathbf{G}_{ji} \ell_j(X_j)\right), \quad \zeta_i(\cdot) = \zeta(u_i, \cdot), \quad \ell_j(\cdot) = \ell(v_j, \cdot), \quad (54.13)$$

where $\zeta(\cdot)$ and $\ell(\cdot)$ are multilayer perceptrons (MLPs) shared across nodes and (u_i, v_j) are trainable node embeddings. A naive implementation would require training $2D$ separate MLPs; weight sharing ensures that f_i depends only on $\mathbf{X}_{\mathbf{Pa}(i; \mathbf{G})}$ while keeping the parameter count modest.

For the exogenous noise $\mathbf{Z}$, DECI supports either (1) Gaussian marginals $p_{Z_i} = \mathcal{N}(0, \sigma_i^2)$ or (2) flexible flows κ_i with standard-normal base distributions [1390]:

$$p_{Z_i}(Z_i) = \mathcal{N}(\kappa_i^{-1}(Z_i); 0, 1) \left| \frac{\partial \kappa_i^{-1}(Z_i)}{\partial Z_i} \right|. \quad (54.14)$$

DECI uses rational-quadratic spline flows [470], parameterised independently across dimensions to preserve noise independence. Although the base distribution is Gaussian, it can be replaced by another distribution with a differentiable density.

Let $\mathbf{x}^1, \dots, \mathbf{x}^N$ denote N observational samples. DECI models these data jointly with the causal graph as

$$p_{\theta}(\mathbf{G}, \mathbf{x}^1, \dots, \mathbf{x}^N) = p(\mathbf{G}) \prod_{n=1}^N p_{\theta}(\mathbf{x}^n \mid \mathbf{G}), \quad (54.15)$$

where the conditional distribution is defined in Equation 54.12. Fitting θ then gives the posterior $p_{\theta}(\mathbf{G} \mid \mathbf{x}^1, \dots, \mathbf{x}^N)$, which characterises uncertainty about the causal structure.

The graph prior $p(\mathbf{G})$ should favour sparse DAGs, but directly representing a distribution over the discrete DAG space is difficult. DECI instead uses the soft DAG constraint $h(\mathbf{G})$ introduced in Equation 54.7 together with the augmented Lagrangian:

$$p(\mathbf{G}) \propto \exp\left(-\lambda_s \|\mathbf{G}\|_F^2 - \alpha h(\mathbf{G}) - \frac{\rho}{2} h(\mathbf{G})^2\right), \quad (54.16)$$

where λ_s controls the sparsity of $\mathbf{G}$, while α and ρ are the augmented-Lagrangian coefficients.

Direct computation of $p_{\theta}(\mathbf{G} \mid \mathbf{x}^1, \dots, \mathbf{x}^N)$ is intractable. DECI therefore uses the mean-field, edge-factorised approximation $q_{\phi}(\mathbf{G})$ with the ENCO parameterisation [1056]:

$$q_{\phi}(\mathbf{G}) = \prod_{i \neq j} \text{Ber}(\mathbf{G}_{ij} \mid \sigma(\gamma_{ij}) \sigma(\vartheta_{ij})), \quad (54.17)$$

where $\sigma(\cdot)$ is the sigmoid function, γ_{ij} represents edge existence, and $\vartheta_{ij} = -\vartheta_{ji}$ represents orientation. The orientation probabilities for $i \rightarrow j$ and $j \rightarrow i$ therefore sum to one, so increasing one decreases the other and helps avoid 2-cycles. Separating existence from orientation also simplifies optimisation of edge probabilities. This

Bernoulli construction is the graph-valued analogue of the mean-field factorisations discussed in Chapter 13. The corresponding evidence lower bound (ELBO) is

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{G})} \left[\log p(\mathbf{G}) + \sum_{n=1}^N \log p_\theta(\mathbf{x}^n | \mathbf{G}) \right] + \mathcal{H}(q_\phi) \leq \log p_\theta(\mathbf{x}^1, \dots, \mathbf{x}^N), \quad (54.18)$$

where $\mathcal{H}(q_\phi)$ is the entropy of q_ϕ . The SEM parameters θ and variational parameters ϕ are trained by maximising the ELBO. The Gumbel–Softmax trick [831] provides stochastic gradient estimates with respect to ϕ .

Statistical guarantees of DECI

The structural-identifiability discussion in Section 54.2.3 motivates a central question for DECI: does maximising the ELBO in Equation 54.18 recover the ground-truth graph and data-generating distribution with infinite data?

The recovery result in Geffner et al. [590] uses the following assumptions:

1. *Non-trivial mechanisms*: each structural mechanism is non-invertible, three-times differentiable, and non-constant in each of its parent inputs.
2. *Regular noise density*: the noise distribution has a differentiable density and bounded log-likelihood.
3. *Correct specification*: there exists a parameter θ^* such that $p_{\theta^*}(\mathbf{X}; \mathbf{G}_0)$ equals the data-generating distribution, where $\mathbf{G}_0$ is the ground-truth graph. The parameter itself need not be unique; recovery is in the distributional or functional sense.
4. *No latent confounders*: the data-generating process satisfies causal sufficiency as defined in Definition 54.3.

The non-constancy and non-invertibility conditions play distinct roles: non-constancy yields causal minimality, whereas non-invertibility excludes non-identifiable functional relationships. Together with the other assumptions, they imply the causal Markov property and structural identifiability while excluding cases such as linear Gaussian ANMs [590].

Theorem 54.1 (DECI recovers the true data-generating process [590]). *Assume that DECI’s SEM is a continuous additive noise model whose mechanisms are non-invertible, three-times differentiable, and non-constant in every parent input, and whose noise density is differentiable with bounded log-likelihood. If the model is correctly specified, the data-generating process has no latent confounders, and infinitely many observations are available, then an ELBO maximiser (θ', q'_ϕ) satisfies*

$$q'_\phi(\mathbf{G}) = \delta_{\mathbf{G}_0}(\mathbf{G}) \quad \text{and} \quad p_{\theta'}(\mathbf{X}; \mathbf{G}_0) = p(\mathbf{X}; \mathbf{G}_0),$$

where $\delta_{\mathbf{G}_0}$ is a point mass at the unique ground-truth graph $\mathbf{G}_0$.

Proof intuition. With infinite data, correct specification and regularity make maximum-likelihood estimation recover the true data-generating distribution. Non-constancy and non-invertibility make the associated graph structurally identifiable, so the maximum-likelihood graph is uniquely $\mathbf{G}_0$. After dividing the ELBO by the sample size, the graph-prior KL term vanishes asymptotically; any graph receiving positive variational mass must therefore attain the same maximum expected log-likelihood. Structural identifiability leaves only $\mathbf{G}_0$, forcing q_ϕ to collapse to a point mass at the true graph [590].

Estimating causal quantities

We now explain how a trained DECI model estimates two causal quantities: the *average treatment effect* (ATE) and *conditional average treatment effect* (CATE). Henceforth, T , Y , and C denote disjoint treatment, outcome, and conditioning index sets, respectively; the corresponding variables are X_T , X_Y , and X_C . The ATE compares two intervention values a and b :

$$\text{ATE}(a, b | \mathbf{G}) = \mathbb{E}[X_Y | \text{do}(X_T=a), \mathbf{G}] - \mathbb{E}[X_Y | \text{do}(X_T=b), \mathbf{G}], \quad (54.19)$$

where $\text{do}(\cdot)$ removes incoming edges to X_T and replaces its structural mechanism by the chosen intervention value [1307, 1318]. The CATE additionally conditions on $X_C = c$:

$$\text{CATE}(a, b | c, \mathbf{G}) = \mathbb{E}[X_Y | \text{do}(X_T=a), X_C=c, \mathbf{G}] - \mathbb{E}[X_Y | \text{do}(X_T=b), X_C=c, \mathbf{G}]. \quad (54.20)$$

As explained above, DECI returns $q_\phi(\mathbf{G})$, an approximation of the posterior over graphs given observational data. Then, interventional distributions, ATE and CATE can be obtained by marginalising over graphs as

$$\mathbb{E}_{q_\phi(\mathbf{G})}[p_\theta(X_Y \mid \text{do}(X_T=a), \mathbf{G})], \quad \mathbb{E}_{q_\phi(\mathbf{G})}[\text{ATE}(a, b \mid \mathbf{G})], \quad \mathbb{E}_{q_\phi(\mathbf{G})}[\text{CATE}(a, b \mid c, \mathbf{G})]. \quad (54.21)$$

This is a probabilistic relaxation of traditional causal estimators. When the posterior is concentrated on one graph, i.e. $q_\phi(\mathbf{G}) = \delta_{\mathbf{G}_i}(\mathbf{G})$, the procedure reduces to causal inference under the fixed graph $\mathbf{G}_i$.

Estimating ATE. After training, DECI can simulate samples $\mathbf{x}$ from $p_\theta(\mathbf{X} \mid \mathbf{G})$ (recall Equation 54.12). We sample a graph $\mathbf{G} \sim q_\phi(\mathbf{G})$ and exogenous noise $\mathbf{z} \sim p_{\mathbf{Z}}$, then apply the ANM in Equation 54.11, using the mechanisms in Equation 54.13, in a topological ordering of $\mathbf{G}$.

However, ATE requires samples from the interventional distribution $p(x_{\setminus T} \mid \text{do}(X_T=b), \mathbf{G})$. These can be obtained from the definition of the do operator:

$$p(x_{\setminus T} \mid \text{do}(X_T=b), \mathbf{G}) = p(x_{\setminus T} \mid X_T=b, \mathbf{G}_{\text{do}(X_T)}),$$

where $\mathbf{G}_{\text{do}(X_T)}$ is the *mutilated* graph obtained by removing incoming edges to X_T . Thus, samples from this distribution are obtained by fixing $X_T = b$ and using $\mathbf{G}_{\text{do}(X_T)}$ instead of $\mathbf{G}$. Their sample mean gives the Monte Carlo estimate required by Equation 54.19. When the posterior $q_\phi(\mathbf{G})$ has not collapsed to one graph, these samples are drawn from a mixture distribution, translating the uncertainty in the discovered graph into uncertainty in ATE estimates.

Estimating CATE. Under the assumption that no variable in X_T is a cause of a variable in X_C , we can estimate CATE by sampling from $p(X_{\setminus T} \mid \text{do}(X_T), \mathbf{G})$ and then estimating the conditional distribution of X_Y given X_C . The conditional density is not directly available because its normalising marginal is intractable. DECI therefore regresses X_Y on X_C using samples simulated under each intervention. Specifically,

$$p_\theta(X_Y \mid \text{do}(X_T=b), X_C=c, \mathbf{G}) = \frac{p_\theta(X_Y, X_C=c \mid X_T=b, \mathbf{G}_{\text{do}(X_T)})}{p_\theta(X_C=c \mid X_T=b, \mathbf{G}_{\text{do}(X_T)})} \quad (54.22)$$

The denominator is not tractable in the DECI model. However, we can sample from the *joint* interventional distribution $p_\theta(X_Y, X_C \mid X_T=b, \mathbf{G}_{\text{do}(X_T)})$.

We use samples from this joint to train a surrogate regression model $h_{\mathbf{G}}^1$ that maps $X_C \mapsto X_Y$ by minimising the squared loss

$$\mathbb{E}_{p_\theta(X_Y, X_C \mid X_T=b, \mathbf{G}_{\text{do}(X_T)})} [\|X_Y - h_{\mathbf{G}}(X_C)\|^2], \quad (54.23)$$

making $h_{\mathbf{G}}$ approximate the conditional mean of X_Y given X_C under the intervention. We choose $h_{\mathbf{G}}$ to be a basis-function linear model with random Fourier features that approximate an RBF kernel. As illustrated in the original DECI exposition, we train two separate surrogate models, one for the intervention $X_T = a$ and one for the reference $X_T = b$. We then estimate CATE as the difference between their outputs evaluated at $X_C = c$. Repeating this for multiple posterior graph samples $\mathbf{G} \sim q_\phi$ yields the graph-marginalised estimate

$$\mathbb{E}_{\mathbf{G} \sim q_\phi(\mathbf{G})} \left[h_{\mathbf{G}_{\text{do}(X_T)}}^{(a)}(c) - h_{\mathbf{G}_{\text{do}(X_T)}}^{(b)}(c) \right].$$

Theoretical guarantees. The recovery result in Section 54.3.1 concerns the true graph and data-generating distribution, not a unique parameter vector. Under correct specification, DECI also satisfies causal-effect identifiability and positivity, so its treatment-effect estimates are correct in the infinite-data limit; Geffner et al. [590] provides the full argument.

DECI provides an end-to-end route from observational data to a flexible nonlinear ANM, a variational posterior over DAGs, and (C)ATE estimates that propagate structural uncertainty. Its edge-factorised variational family can nevertheless be limiting for complicated graph posteriors. Moreover, acyclicity is enforced *softly*, so non-DAGs can be sampled before the augmented-Lagrangian optimisation converges. The next section introduces *BayesDAG*, a sampling-based approach using a node-potential parameterisation that produces DAGs throughout training.

54.3.2 BayesDAG: Fully Bayesian treatment through sampling

The DECI framework already brings a Bayesian perspective to causal discovery: it learns a variational posterior over DAGs and propagates structural uncertainty into interventional estimates. Its edge-factorised posterior and soft acyclicity constraint, however, motivate an alternative treatment of the graph space. *BayesDAG* keeps the same downstream causal-inference recipe but replaces the discovery component with an acyclicity-by-construction

¹The subscript $\mathbf{G}$ distinguishes surrogate models fitted to samples from different graphs drawn from q_ϕ .

parameterisation and a stochastic-gradient Markov chain Monte Carlo (SG-MCMC) scheme. Concretely, it represents DAGs through the node potentials introduced in Section 54.2.5 and a binary edge mask, and then connects this representation to permutation learning to obtain useful gradients for the potentials. SG-MCMC, introduced in Chapter 2, is then used to sample the continuous node potentials and SEM parameters, while variational inference handles the discrete mask. Once samples from the joint graph-and-model posterior are available, causal quantities are estimated using the graph-marginalised simulation approach introduced for DECI. An implementation of BayesDAG is available at <https://github.com/microsoft/Project-BayesDAG>.

Bayesian inference in the DAG space

To guarantee that every sampled graph is a DAG, BayesDAG reparameterises the graph space so that acyclicity is *built in* rather than enforced by a smooth penalty as in DECI. Recall the weighted NoCurl map in Equation 54.10, which uses $\text{ReLU}(\text{grad } \mathbf{p})$ to orient continuous edge weights. For posterior inference over *binary* DAGs, this form has two drawbacks: (i) the potential-based orientation permits one edge for every unordered pair of nodes, so the continuous entries of $\mathbf{W}$ must be driven exactly to zero to disable edges, and (ii) the same continuous weights encode both edge existence and strength, entangling structure with parameter learning. BayesDAG separates these roles by replacing the weighted ReLU construction with the binary map

$$\tau : \{0, 1\}^{D \times D} \times \mathbb{R}^D \longrightarrow \{0, 1\}^{D \times D}, \quad \tau(\mathbf{W}, \mathbf{p}) = \mathbf{W} \odot \text{Step}(\text{grad } \mathbf{p}), \quad (54.24)$$

where $\text{Step}(\cdot)$ is applied elementwise. Here, $\mathbf{W}$ is a *binary* edge mask that encodes existence and controls the graph sparsity, while $\mathbf{p}$ orients those edges. By construction, $\tau(\mathbf{W}, \mathbf{p})$ is a binary adjacency matrix with no directed cycles.

Equivalence of Bayesian inference. Let $\mathcal{G}_D$ be the set of binary DAG adjacencies on D nodes and let $p(\mathbf{G})$ be any prior on $\mathcal{G}_D$. The core idea of BayesDAG is to perform inference in the product space $(\mathbf{W}, \mathbf{p}) \in \{0, 1\}^{D \times D} \times \mathbb{R}^D$ and push the resulting posterior forward through τ . The following result [35] formalises the equivalence, provided that the prior over $(\mathbf{W}, \mathbf{p})$ induces $p(\mathbf{G})$ and ties among the potentials are excluded.

Theorem 54.2 (Equivalence of inference in $(\mathbf{W}, \mathbf{p})$ and binary DAG space [35]). *Assume that the entries of $\mathbf{p}$ are pairwise distinct. Let $p_\tau(\mathbf{W}, \mathbf{p})$ be a joint prior mass-density with respect to counting measure on binary $\mathbf{W}$ and Lebesgue measure on continuous $\mathbf{p}$. If this prior induces the desired graph prior,*

$$p(\mathbf{G}) = \sum_{\mathbf{W} \in \{0, 1\}^{D \times D}} \int_{\mathbb{R}^D} p_\tau(\mathbf{W}, \mathbf{p}) \mathbf{1}\{\mathbf{G} = \tau(\mathbf{W}, \mathbf{p})\} d\mathbf{p},$$

then its posterior induces the corresponding graph posterior:

$$p(\mathbf{G} \mid \mathbf{D}) = \sum_{\mathbf{W} \in \{0, 1\}^{D \times D}} \int_{\mathbb{R}^D} p_\tau(\mathbf{W}, \mathbf{p} \mid \mathbf{D}) \mathbf{1}\{\mathbf{G} = \tau(\mathbf{W}, \mathbf{p})\} d\mathbf{p}. \quad (54.25)$$

Proof intuition. The map τ is complete: every pair $(\mathbf{W}, \mathbf{p})$ produces a DAG, and every DAG can be recovered by choosing potentials consistent with one of its topological orderings and setting $\mathbf{W}$ to its edge mask. The prior condition above therefore says that $p_\tau(\mathbf{W}, \mathbf{p})$ pushes forward to $p(\mathbf{G})$. For a fixed graph, the likelihood depends on $(\mathbf{W}, \mathbf{p})$ only through $\tau(\mathbf{W}, \mathbf{p}) = \mathbf{G}$; applying Bayes' rule and summing and integrating over all pairs in this preimage consequently yields $p(\mathbf{D} \mid \mathbf{G})p(\mathbf{G})$ divided by the same evidence as direct inference in $\mathcal{G}_D$. Hence, the posterior pushforward in Equation 54.25 is exactly $p(\mathbf{G} \mid \mathbf{D})$.

Model and posterior in the augmented space. Let Θ denote the SEM parameters, such as the parameters of the nonlinear functions introduced for DECI. BayesDAG reuses the standard ANM likelihood in Equation 54.12, but conditions it on $\mathbf{G} = \tau(\mathbf{W}, \mathbf{p})$:

$$p(\mathbf{D}, \mathbf{W}, \mathbf{p}, \Theta) = p(\mathbf{D} \mid \mathbf{G} = \tau(\mathbf{W}, \mathbf{p}), \Theta) p_\tau(\mathbf{W}, \mathbf{p}) p(\Theta).$$

A convenient choice places a Gaussian prior on $\mathbf{p}$ and factorised Bernoulli priors on the entries of $\mathbf{W}$, thereby controlling expected sparsity. The resulting posterior target is

$$p_\tau(\mathbf{W}, \mathbf{p}, \Theta \mid \mathbf{D}) \propto p(\mathbf{D} \mid \tau(\mathbf{W}, \mathbf{p}), \Theta) p_\tau(\mathbf{W}, \mathbf{p}) p(\Theta). \quad (54.26)$$

The continuous variables $\mathbf{p}$ and Θ can be sampled with SG-MCMC without a Metropolis–Hastings acceptance step. However, the piecewise-constant $\text{Step}(\cdot)$ in τ provides no useful derivative with respect to $\mathbf{p}$, so it cannot be used directly in a gradient-based sampler.

Equivalent formulation: enabling gradients for node potentials

Intuition. The node potentials $\mathbf{p}$ define a topological order via pairwise differences: $\text{Step}((\text{grad } \mathbf{p})(i, j))$ keeps only downhill orientations. Thus, ranking nodes by their potentials yields a permutation that orients all admissible edges from higher to lower potential. This observation lets us replace the non-differentiable $\text{Step}(\cdot)$ with a permutation-learning view.

Alternative formulation. Let $\mathbf{L} \in \{0, 1\}^{D \times D}$ be the strictly lower-triangular matrix of ones and $\mathbf{o} = [1, \dots, D]^\top$. Define

$$\mathbf{G} = \mathbf{W} \odot [\text{perm}(\mathbf{p}) \mathbf{L} \text{perm}(\mathbf{p})^\top], \quad \text{where } \text{perm}(\mathbf{p}) = \arg \max_{\mathbf{\Pi} \in \Sigma_D} \mathbf{p}^\top (\mathbf{\Pi} \mathbf{o}). \quad (54.27)$$

Here, Σ_D is the set of $D \times D$ permutation matrices. The permutation $\text{perm}(\mathbf{p})$ encodes the ranking induced by the potentials, $\text{perm}(\mathbf{p}) \mathbf{L} \text{perm}(\mathbf{p})^\top$ orients all possible edges according to that ranking, and the mask $\mathbf{W}$ removes absent edges. The following result shows that this construction coincides with Equation 54.24.

Theorem 54.3 (Equivalence to original formulation [35]). *Assume that the conditions of Theorem 54.2 hold. Then, for a given $(\mathbf{W}, \mathbf{p})$,*

$$\mathbf{G} = \mathbf{W} \odot \text{Step}(\text{grad } \mathbf{p}) = \mathbf{W} \odot [\text{perm}(\mathbf{p}) \mathbf{L} \text{perm}(\mathbf{p})^\top],$$

where $\mathbf{G}$ is a DAG and $\text{perm}(\mathbf{p})$ is defined in Equation 54.27.

This theorem translates the original formulation into finding a corresponding permutation $\text{perm}(\mathbf{p})$. Although this does not itself resolve the lack of an informative gradient, it allows the permutation to be approximated using tools from the differentiable-permutation literature [171, 1162]. BayesDAG uses the standard Sinkhorn relaxation [1162], which replaces a discrete assignment with a differentiable doubly-stochastic approximation; other differentiable permutation estimators could be used instead.

Sinkhorn operator. The argmax over permutations in Equation 54.27 is non-differentiable. Let $\text{Sinkhorn}(\mathbf{M})$ denote the operator that iteratively normalises the rows and columns of a matrix $\mathbf{M}$ to obtain a doubly-stochastic matrix. This entropy-regularised relaxation of the assignment problem is a standard differentiable surrogate widely used in optimal transport and learning-to-rank [351], and is not specific to the causal setting. The generic assignment problem

$$\mathbf{\Pi}^* = \arg \max_{\mathbf{\Pi} \in \Sigma_D} \langle \mathbf{\Pi}, \mathbf{M} \rangle, \quad (54.28)$$

admits an entropy-regularised relaxation whose solution is expressed by the Sinkhorn operator [1162]:

$$\text{Sinkhorn}(\mathbf{M}/t) = \arg \max_{\mathbf{\Pi} \in \mathcal{B}_D} \left\{ \langle \mathbf{\Pi}, \mathbf{M} \rangle + t h(\mathbf{\Pi}) \right\}, \quad \lim_{t \rightarrow 0} \text{Sinkhorn}(\mathbf{M}/t) \in \Sigma_D,$$

where $\mathcal{B}_D$ is the Birkhoff polytope of $D \times D$ doubly-stochastic matrices and $h(\cdot)$ is the entropy function. Applying this relaxation to Equation 54.27 with the rank-1 score matrix $\mathbf{M} = \mathbf{p} \mathbf{o}^\top$ gives

$$\arg \max_{\mathbf{\Pi} \in \Sigma_D} \mathbf{p}^\top (\mathbf{\Pi} \mathbf{o}) = \arg \max_{\mathbf{\Pi} \in \Sigma_D} \langle \mathbf{\Pi}, \mathbf{p} \mathbf{o}^\top \rangle = \lim_{t \rightarrow 0} \text{Sinkhorn} \left(\frac{\mathbf{p} \mathbf{o}^\top}{t} \right). \quad (54.29)$$

In practice, a small positive temperature t yields differentiable doubly-stochastic scores. The Hungarian algorithm [1210] projects these scores to a binary permutation for the forward computation, while a straight-through estimator backpropagates through the Sinkhorn relaxation. This provides useful approximate gradients for the node potentials $\mathbf{p}$.

Bayesian causal discovery via sampling

The preceding relaxation provides approximate gradients of the reparameterised graph posterior with respect to $\mathbf{p}$. BayesDAG uses these gradients in a combined scheme: SG-MCMC samples the continuous variables $(\mathbf{p}, \boldsymbol{\Theta})$, while a lightweight variational update handles the discrete mask $\mathbf{W}$.

For its nonlinear SEM, BayesDAG uses the same ANM likelihood as DECI in Section 54.3.1:

$$p(\mathbf{x} \mid \mathbf{G}, \boldsymbol{\Theta}) = p_\epsilon(\mathbf{x} - \mathbf{f}(\mathbf{x}; \boldsymbol{\Theta})) = \prod_{i=1}^D p_{\epsilon_i}(x_i - f_i(\mathbf{x}_{\mathbf{Pa}(i, \mathbf{G})})). \quad (54.30)$$

Prior design. The graph prior $p(\mathbf{G})$ is induced through $p_\tau(\mathbf{W}, \mathbf{p})$. Let $p(\mathbf{W}) = \prod_{i \neq j} \text{Ber}(W_{ij} \mid \pi_{ij})$ be a factorised Bernoulli baseline prior. One convenient joint prior is

$$p(\mathbf{W}, \mathbf{p}, \Theta) \propto \mathcal{N}(\Theta; \mathbf{0}, \mathbf{I}) \mathcal{N}(\mathbf{p}; \mathbf{0}, \alpha \mathbf{I}) p(\mathbf{W}) \exp(-\lambda_s \|\tau(\mathbf{W}, \mathbf{p})\|_F^2), \quad (54.31)$$

where α controls the prior scale of $\mathbf{p}$, the probabilities π_{ij} control baseline edge inclusion, and λ_s further encourages sparsity in $\mathbf{G}$.

The main remaining challenge is the binary nature of $\mathbf{W} \in \{0, 1\}^{D \times D}$. BayesDAG addresses it by combining SG-MCMC for $\mathbf{p}, \Theta$ with variational inference for $\mathbf{W}$. Figure 54.2 summarises the dependencies used in this hybrid inference scheme.

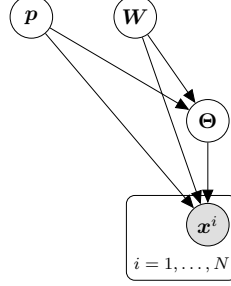


Figure 54.2: Graphical model of the BayesDAG inference formulation. Here, $\mathbf{p}$ and $\mathbf{W}$ denote the node potentials and binary edge mask, Θ denotes the SEM parameters, and $\mathbf{x}^i$ denotes observed sample i .

The Gibbs-style procedure alternates between (1) sampling $\mathbf{p}, \Theta \sim p(\mathbf{p}, \Theta \mid \mathbf{D}, \mathbf{W})$ with SG-MCMC and (2) updating the variational posterior $q_\phi(\mathbf{W} \mid \mathbf{p})$ to approximate $p(\mathbf{W} \mid \mathbf{p}, \Theta, \mathbf{D})$.

The following proposition specifies the gradients required by SG-MCMC, where $U(\mathbf{p}, \mathbf{W}, \Theta) = -\log p(\mathbf{D}, \mathbf{W}, \Theta, \mathbf{p})$.

Proposition 54.1 ([35]). *For the model above, the required gradients are*

$$\nabla_{\mathbf{p}} U = -\nabla_{\mathbf{p}} \log p(\mathbf{p}) - \nabla_{\mathbf{p}} \log p(\mathbf{D} \mid \Theta, \tau(\mathbf{W}, \mathbf{p})) \quad (54.32)$$

and

$$\nabla_{\Theta} U = -\nabla_{\Theta} \log p(\Theta) - \nabla_{\Theta} \log p(\mathbf{D} \mid \Theta, \tau(\mathbf{W}, \mathbf{p})). \quad (54.33)$$

For $\mathbf{W}$, the variational posterior $q_\phi(\mathbf{W} \mid \mathbf{p})$ approximates $p(\mathbf{W} \mid \mathbf{p}, \Theta, \mathbf{D})$. Specifically, a neural network $\mu_\phi(\mathbf{p})$ maps the node potentials to edge logits, which are converted to independent Bernoulli probabilities:

$$q_\phi(\mathbf{W} \mid \mathbf{p}) = \prod_{i \neq j} \text{Ber}(W_{ij} \mid \sigma(\mu_\phi(\mathbf{p})_{ij})). \quad (54.34)$$

Similar to DECI, we can derive the corresponding *evidence lower bound* (ELBO) to train the approximate posterior.

$$\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(\mathbf{W} \mid \mathbf{p})} [\log p(\mathbf{D}, \mathbf{p}, \Theta \mid \mathbf{W})] - \text{D}_{\text{KL}}[q_\phi(\mathbf{W} \mid \mathbf{p}) \parallel p(\mathbf{W})]. \quad (54.35)$$

The following algorithm describes the procedure of BayesDAG:

BayesDAG extends DECI's Bayesian treatment of static causal discovery, but many real-world systems are temporal, a setting that neither method handles. The next section adapts these causal-discovery ideas to temporal models while retaining structural-identifiability guarantees.

54.4 Bayesian Deep Learning for Causality with Temporal Data

The causal mechanisms underlying many real-world systems evolve over time. In domains such as climate science, finance, neuroscience, and healthcare, observations arrive as multivariate time series, and causal influence may act both within and across time steps. Learning these relations from observational data introduces three challenges beyond the static setting of Section 54.3: mechanisms may be nonlinear, effects may be instantaneous at the sampling resolution, and the noise distribution may depend on the system's history.

Algorithm 38: BayesDAG: SG-MCMC + VI Inference

Input : dataset $\mathbf{D}$; priors $p_\tau(\mathbf{W}, \mathbf{p})$, $p(\boldsymbol{\Theta})$; SG-MCMC sampler $\mathcal{S}$; sampler hyperparameters Ψ ; neural network $\mu_\phi(\cdot)$; training iterations T

Output: samples $\{\boldsymbol{\Theta}, \mathbf{p}\}$ and variational posterior q_ϕ

initialize $\boldsymbol{\Theta}^{(0)}, \mathbf{p}^{(0)}, \phi$

for $t = 1$ **to** T **do**

- sample $\mathbf{W}^{(t-1)} \sim q_\phi(\mathbf{W} \mid \mathbf{p}^{(t-1)})$
- evaluate $\nabla_{\mathbf{p}, \boldsymbol{\Theta}} U$ (Equations 54.32 and 54.33) with $\boldsymbol{\Theta}^{(t-1)}, \mathbf{p}^{(t-1)}, \mathbf{W}^{(t-1)}$
- $\boldsymbol{\Theta}^{(t)}, \mathbf{p}^{(t)} \leftarrow \mathcal{S}(\nabla_{\mathbf{p}, \boldsymbol{\Theta}} U; \Psi)$
- if** *storing condition met* **then**
 - append $\{\mathbf{p}^{(t)}, \boldsymbol{\Theta}^{(t)}\}$ to samples
- maximise the ELBO (Equation 54.35) with respect to ϕ using $\mathbf{p}^{(t)}, \boldsymbol{\Theta}^{(t)}$

Classical *Granger causality* [658] identifies lagged predictive dependencies by asking whether one series' past improves prediction of another. Classical Granger models focus on lagged effects and commonly rely on linearity and stationarity, so they do not directly represent instantaneous structure or history-dependent noise. DYNOTEARS [1285] adds a differentially learned instantaneous DAG to a vector-autoregressive model, but retains linear mechanisms and additive, history-independent errors. *Rhino* [647] addresses all three challenges with a nonlinear temporal SEM and a variational posterior over graph structures.

Continuous-time methods address a complementary difficulty: irregular observations may not admit a natural fixed sampling interval. After developing discrete-time *Rhino*, we therefore briefly introduce *SCOTCH* [1701], which combines a latent stochastic differential equation (SDE) with variational inference over causal graphs and latent trajectories.

54.4.1 Rhino: Deep Causal Temporal Relationship Learning with History-Dependent Noise

In multivariate time series, coarse sampling can aggregate events between observations and make their effects appear instantaneous. At the same time, the magnitude and shape of the noise may change with the past. For example, an announcement made after a stock market closes may have a nonlinear, apparently instantaneous effect at reopening and may also change subsequent volatility. *Rhino* extends DECI to this setting by combining flexible neural mechanisms, a temporal causal graph, and a conditional noise model whose distribution depends on lagged observations. It is trained with a DECI-style variational objective and retains structural-identifiability guarantees in this history-dependent regime.

Rhino model formulation

We first extend the static SEM notation to the temporal regime. Let $\{\mathbf{X}_t\}_{t=0}^T$ be a discrete-time, D -variate process with $\mathbf{X}_t = (X_t^1, \dots, X_t^D)^\top$, where T is the final sequence index. Let K be the maximum lag and let $\mathbf{G}_{0:K} = \{\mathbf{G}_\tau\}_{\tau=0}^K$ denote the stationary temporal-graph template. For $\tau \geq 1$, $(\mathbf{G}_\tau)_{ji} = 1$ encodes the lagged edge $X_{t-\tau}^j \rightarrow X_t^i$, while $\mathbf{G}_0$ collects instantaneous edges among $\{X_t^i\}_{i=1}^D$ at a fixed time. Denote the corresponding parent sets by

$$\mathbf{Pa}_{\mathbf{G}}^i(< t) = \{X_{t-\tau}^j : (\mathbf{G}_\tau)_{ji} = 1, \tau \in \{1, \dots, K\}\}, \quad \mathbf{Pa}_{\mathbf{G}}^i(t) = \{X_t^j : (\mathbf{G}_0)_{ji} = 1\},$$

a temporal SEM specifies, for each i and t ,

$$X_t^i = f_{i,t}(\mathbf{Pa}_{\mathbf{G}}^i(< t), \mathbf{Pa}_{\mathbf{G}}^i(t), \epsilon_t^i), \quad (54.36)$$

with mutually independent exogenous noises ϵ_t^i across nodes and times. The mechanism $f_{i,t}$ respects the template: its derivative with respect to $X_{t-\tau}^j$ is zero whenever $(\mathbf{G}_\tau)_{ji} = 0$. Under causal stationarity, the mechanisms and $\mathbf{G}_{0:K}$ do not change with t . The template $\mathbf{G}_{0:K}$ should not be confused with the *time-unrolled graph*: instantiating it over $t = 0, \dots, T$ produces a graph on the individual nodes $\{X_t^i\}_{i,t}$. Lagged edges in this unrolled graph always point forward in time, and the full graph is a DAG provided that the instantaneous template $\mathbf{G}_0$ is itself a DAG. A linear special case is the structural vector autoregressive (VAR) model

$$X_t^i = \beta^i + \sum_{\tau=0}^K \sum_{j=1}^D B_{\tau,ji} X_{t-\tau}^j + \epsilon_t^i, \quad (54.37)$$

where $\mathbf{B}_\tau$ is the weighted adjacency matrix at lag τ . Its structural identifiability follows under assumptions such as non-Gaussian errors or equal error variances [808, 1285], as in the static setting.

Rhino extends the DECI model in Section 54.3.1 to this temporal setting. Its structural equation model retains an additive decomposition while allowing nonlinear dependencies, instantaneous effects, and history-dependent noise:

$$X_t^i = f_i(\mathbf{Pa}_G^i(< t), \mathbf{Pa}_G^i(t)) + g_i(\mathbf{Pa}_G^i(< t), \epsilon_t^i), \quad (54.38)$$

where f_i is a differentiable nonlinear causal mechanism and g_i is a differentiable, history-dependent transform that is invertible in its second argument, ϵ_t^i . Compared with the additive noise models used in DECI and other temporal SEMs [1285, 1317], this formulation allows the conditional noise distribution itself to vary with the lagged parents.

The masks enforce the template: f_i may depend only on its lagged and instantaneous parents, whereas g_i may depend only on lagged parents. In practice, f_i uses a masked, weight-sharing neural architecture similar to Equation 54.13:

$$f_i(\mathbf{Pa}_G^i(< t), \mathbf{Pa}_G^i(t)) = \zeta_i \left(\sum_{\tau=0}^K \sum_{j=1}^D G_{\tau,ji} \ell_{\tau j}(X_{t-\tau}^j) \right), \quad (54.39)$$

where ζ_i and $\ell_{\tau j}$ are neural networks. For efficiency, weights are shared across nodes and lags through $\zeta_i(\cdot) = \zeta(\cdot, \mathbf{u}_{0,i})$ and $\ell_{\tau j}(\cdot) = \ell(\cdot, \mathbf{u}_{\tau,j})$, with trainable embeddings $\mathbf{u}_{\tau,i}$ for node i at lag τ . This generalises DECI's weight-sharing parameterisation to temporal graphs.

The design of g_i must balance flexibility with tractability of the likelihood. Rhino uses a *conditional spline flow* [470, 1304, 1635], with base Gaussian noise ϵ_t^i for all t and i . The spline parameters are predicted by a hypernetwork whose inputs are the lagged parents $\mathbf{Pa}_G^i(< t)$, again in the masked, weight-sharing style of (54.39) but with $\tau \in [1, K]$ only (instantaneous parents excluded). Because g_i is invertible in its second argument, the noise likelihood conditional on lagged parents is tractable:

$$p_{g_i}(z_t^i | \mathbf{Pa}_G^i(< t)) = p_\epsilon(g_i^{-1}(\mathbf{Pa}_G^i(< t), z_t^i)) \left| \frac{\partial g_i^{-1}(\mathbf{Pa}_G^i(< t), z_t^i)}{\partial z_t^i} \right|, \quad (54.40)$$

where $z_t^i = g_i(\mathbf{Pa}_G^i(< t), \epsilon_t^i)$ and p_ϵ is the base Gaussian density. Thus, the conditional distribution of X_t^i can depend flexibly on the system's history while retaining a tractable likelihood.

Variational inference for Rhino

Rhino adopts the same Bayesian perspective as DECI: rather than outputting a single graph, it approximates a posterior over temporal-graph templates using the variational framework reviewed in Chapter 12. For N observed multivariate time series $\{\mathbf{X}_{0:T}^{(1)}, \dots, \mathbf{X}_{0:T}^{(N)}\}$ and model parameters θ , the joint distribution is

$$p(\mathbf{X}_{0:T}^{(1)}, \dots, \mathbf{X}_{0:T}^{(N)}, \mathbf{G}_{0:K}) = p(\mathbf{G}_{0:K}) \prod_{n=1}^N p_\theta(\mathbf{X}_{0:T}^{(n)} | \mathbf{G}_{0:K}). \quad (54.41)$$

Once fitted, $p(\mathbf{G}_{0:K} | \mathbf{X}_{0:T}^{(1)}, \dots, \mathbf{X}_{0:T}^{(N)})$ quantifies uncertainty over lagged and instantaneous causal relationships.

Graph prior. The design of $p(\mathbf{G}_{0:K})$ balances structural constraints with flexibility. Following the smooth DAG formulation used in DECI, we combine three ingredients: (i) a DAG constraint, (ii) a sparsity prior, and (iii) optional domain-specific information:

$$p(\mathbf{G}_{0:K}) \propto \exp\left(-\lambda_s \|\mathbf{G}_{0:K}\|_F^2 - \rho h^2(\mathbf{G}_0) - \alpha h(\mathbf{G}_0) - \lambda_p \|\mathbf{G}_{0:K} - \mathbf{G}_{0:K}^p\|_F^2\right), \quad (54.42)$$

where $h(\mathbf{G}_0) = \text{tr}(e^{\mathbf{G}_0 \odot \mathbf{G}_0}) - D$ is the smooth DAG penalty and $\mathbf{G}_{0:K}^p$ is an optional prior template encoding domain knowledge. The lagged matrices $\mathbf{G}_{1:K}$ automatically respect temporal direction in the unrolled graph, so the acyclicity penalty applies only to $\mathbf{G}_0$. Here, λ_s and λ_p control sparsity and prior strength, while $\alpha \geq 0$ and $\rho > 0$ are the augmented-Lagrangian coefficients updated as in DECI.

Variational objective. As in DECI, the exact posterior over $\mathbf{G}_{0:K}$ is intractable. Rhino therefore uses an edge-factorised Bernoulli approximation $q_\phi(\mathbf{G}_{0:K})$ and maximises the corresponding evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{X}_{0:T}^{(1)}, \dots, \mathbf{X}_{0:T}^{(N)}) \geq \underbrace{\mathbb{E}_{q_\phi(\mathbf{G}_{0:K})} \left[\sum_{n=1}^N \log p_\theta(\mathbf{X}_{0:T}^{(n)} | \mathbf{G}_{0:K}) + \log p(\mathbf{G}_{0:K}) \right]}_{\text{ELBO}(\theta, \phi)} + H(q_\phi(\mathbf{G}_{0:K})), \quad (54.43)$$

where $H(\cdot)$ denotes entropy.

From the causal Markov assumption and autoregressive structure of Rhino, the sequence likelihood factorises as

$$\log p_\theta(\mathbf{X}_{0:T}^{(n)} | \mathbf{G}_{0:K}) = \sum_{t=0}^T \sum_{i=1}^D \log p_\theta(X_t^{i,(n)} | \mathbf{Pa}_\mathbf{G}^i(< t), \mathbf{Pa}_\mathbf{G}^i(t)), \quad (54.44)$$

and from Rhino’s SEM (Equation 54.38) we obtain

$$\log p_\theta(X_t^{i,(n)} | \mathbf{Pa}_\mathbf{G}^i(< t), \mathbf{Pa}_\mathbf{G}^i(t)) = \log p_{g_i}(z_t^{i,(n)} | \mathbf{Pa}_\mathbf{G}^i(< t)), \quad (54.45)$$

where $z_t^{i,(n)} = X_t^{i,(n)} - f_i(\mathbf{Pa}_\mathbf{G}^i(< t), \mathbf{Pa}_\mathbf{G}^i(t))$ and p_{g_i} is the history-conditioned flow density defined in Equation 54.40. The parameters θ and ϕ are learned by maximising this ELBO with a Gumbel–Softmax gradient estimator and the same augmented-Lagrangian updates as DECI, driving the instantaneous subgraph $\mathbf{G}_0$ towards a DAG. Once trained, Rhino provides a variational posterior over the stationary template $\mathbf{G}_{0:K}$. Each template sample specifies both lagged and instantaneous relations and induces a time-unrolled graph over the observed sequence of length $T + 1$. Because Rhino also learns the functional mechanisms, it supports temporal causal inference using the simulation strategy introduced in Section 54.3.1.

Structural identifiability of Rhino

Like DECI’s recovery result in Section 54.3.1, Rhino provides two guarantees: (i) structural identifiability of the temporal SEM in (54.38)–(54.40), and (ii) validity of the variational objective used for discovery. Under the stated assumptions, these results recover the true temporal-graph template and data-generating distribution in the infinite-data limit. The full proofs are given by Gong et al. [647].

The central challenge is to ensure that two distinct temporal templates cannot induce the same observational distribution.

Theorem 54.4 (Identifiability of Rhino SEMs [647]). *Assume that the stationary SEM in (54.38) satisfies the causal Markov property, causal minimality, causal sufficiency, acyclicity of $\mathbf{G}_0$, and induces a proper joint density. Suppose further that: (1) all functions and induced densities are third-order differentiable; (2) f_i is nonlinear and, for every instantaneous parent $X_t^j \in \mathbf{Pa}_\mathbf{G}^i(t)$, its dependence on X_t^j with the other inputs fixed is non-invertible; and (3) the second derivative of $\log p_{g_i}(g_i(\epsilon_t^i) | \mathbf{Pa}_\mathbf{G}^i(< t))$ with respect to ϵ_t^i is zero only at isolated points. Then the SEM in (54.38) is structurally identifiable for both bivariate and multivariate time series.*

Proof logic. The proof proceeds in three steps. First, an identifiable source model fixes the initial segment, while the causal Markov and minimality assumptions distinguish templates that disagree only on lagged parents. Conditioning on the observed history then reduces the remaining directional ambiguity to the instantaneous parent–child relations. Second, Rhino is embedded in a history-dependent post-nonlinear class; after applying its invertible outer transformations, the conditional bivariate problem reduces to an additive-noise argument of the type used by Peters et al. [1317]. The differentiability and density conditions exclude the exceptional cases in which both instantaneous directions generate the same conditional distribution. Finally, the pairwise result is lifted to the multivariate setting using identifiable functional model class arguments [1316], with the bivariate condition required for each instantaneous parent.

The standard history-independent ANM is recovered by setting $g_i(\mathbf{Pa}_\mathbf{G}^i(< t), \epsilon_t^i) = \epsilon_t^i$. Rhino therefore extends the static ANM result: the restriction above identifies each instantaneous direction, while the invertible transform g_i permits the noise distribution to vary with lagged history.

Validity of variational objective

Rhino’s variational objective mirrors DECI’s ELBO. Under correct specification and the assumptions of Theorem 54.4, maximising it recovers the true temporal template in the infinite-data limit.

Theorem 54.5 (Validity of the variational objective [647]). *Under the conditions of Theorem 54.4 and assuming no model misspecification, any maximiser $(\theta', q'_\phi(\mathbf{G}_{0:K}))$ of the ELBO in (54.43) with infinite data satisfies*

$$q'_\phi(\mathbf{G}_{0:K}) = \delta(\mathbf{G}_{0:K} = \mathbf{G}_{0:K}^*) \quad \text{and} \quad p_{\theta'}(\mathbf{X}_{0:T} \mid \mathbf{G}_{0:K}^*) = p(\mathbf{X}_{0:T} \mid \mathbf{G}_{0:K}^*),$$

where $\mathbf{G}_{0:K}^*$ is the ground-truth temporal template and the right-hand side is the true data-generating distribution.

Rhino thus extends DECI to discrete-time systems with lagged and instantaneous effects and history-dependent noise. Its fixed temporal grid, however, is unsuitable when observations arrive irregularly or the underlying dynamics are intrinsically continuous.

54.4.2 Continuous-time extension of Rhino: SCOTCH

Physical, financial, and epidemiological systems are often observed at irregular intervals, for which imposing a fixed discrete-time grid can obscure the underlying dynamics or induce spurious dependencies. Neural ordinary differential equation methods such as NGM [113, 272] model a deterministic mean process and extract its graph from a neural network, but NGM assumes a single observed series with a multivariate Gaussian law. This restricts it to comparatively simple stochastic dynamics.

SCOTCH [1701] instead models intrinsic process noise with a latent SDE and separates it from observation noise:

$$\begin{aligned} d\mathbf{Z}_t &= \mathbf{f}_\theta(\mathbf{Z}_t, \mathbf{G}) dt + \mathbf{g}_\theta(\mathbf{Z}_t, \mathbf{G}) d\mathbf{W}_t, \\ \mathbf{X}_{t_i} &= \mathbf{Z}_{t_i} + \boldsymbol{\epsilon}_{t_i}, \end{aligned}$$

where $\mathbf{Z}_t$ is the latent state, $\mathbf{W}_t$ is Brownian motion, and $\boldsymbol{\epsilon}_{t_i}$ is independent measurement noise. The graph-masked drift $\mathbf{f}_\theta$ captures deterministic dependencies, while the diagonal diffusion $\mathbf{g}_\theta$ captures state-dependent stochasticity. With no instantaneous effects, an Euler discretisation of this model has the same residual form as Rhino, making SCOTCH its continuous-time analogue.

SCOTCH uses variational inference to approximate both the graph posterior $q_\phi(\mathbf{G})$ and, for each observed series, a posterior SDE over latent trajectories. Its ELBO combines the expected observation likelihood, a path-space term comparing the posterior and prior dynamics, and the graph divergence $D_{\text{KL}}[q_\phi(\mathbf{G}) \parallel p(\mathbf{G})]$. This extends the graph-level variational machinery of Chapter 12 to continuous latent paths.

Together, the four methods in this chapter illustrate complementary uses of Bayesian deep learning for causality: DECI and BayesDAG address static discovery, Rhino extends discovery to discrete-time systems with history-dependent noise, and SCOTCH handles continuous-time latent dynamics and irregular observations.

54.5 Further advances in causal modelling

The preceding sections focused on Bayesian causal discovery and inference for static, discrete-time, and continuous-time data. Causal reasoning is also increasingly relevant in the era of foundation models: it underpins the ability to anticipate the consequences of actions, which lies at the core of *world modelling*. We conclude with two complementary directions that use transformer architectures to scale causal modelling. The first recasts SCM generation as a fixed-point problem on causally ordered variables [1466]; the second amortises treatment-effect estimation across datasets [1848]. We then close the section with pointers to other active research relating to causal foundation models. These examples illustrate recent advances rather than providing a comprehensive survey.

Fixed-point causal generative modelling. DECI and BayesDAG represent the causal graph explicitly, through a variational distribution over adjacency matrices or a node-potential parameterisation. Fixed-point causal models take a different route. Given a topological ordering represented by a permutation P , they define a joint mechanism $\mathbf{H}$ on the causally ordered variables and their exogenous noise. Under suitable architectural conditions, solving the resulting fixed-point problem generates the same observations as the corresponding SCM. The graph therefore need not be explicitly constructed or regularised while learning the mechanisms, although a valid causal ordering is still required.

The fixed-point approach, called FiP, separates learning into two stages: (i) an amortised model predicts a valid topological ordering from observational data, and (ii) a transformer-based model learns $\mathbf{H}$ conditional on that ordering. Its attention architecture enforces a lower-triangular dependency pattern, so each variable can depend only on variables preceding it in the proposed order. This removes the need for an acyclicity penalty in the

mechanism-learning stage. The fixed-point formalism is equivalent to a classical SCM, and Scetbon et al. [1466] establish conditions for unique recovery of the SCM when the topological ordering is given.

Causal inference with attention. CInA (*Causal Inference with Attention*) instead addresses treatment-effect estimation. Its starting point is optimal covariate balancing: non-negative weights are assigned to observational units so that the weighted covariate distributions of the treated and control groups are similar. Under the usual causal-identification assumptions, differences between weighted outcomes can then estimate the average treatment effect (ATE). Such weights can be obtained through a convex support-vector-machine (SVM) formulation [1591].

Zhang et al. [1848] establish a primal–dual connection between this optimal-balancing objective and self-attention, and use it to design CInA. The interpretation is specific to their transformer parameterisation, training objective, and causal assumptions: it is not an inherent property of arbitrary attention weights. Quantities produced by CInA’s final layer are transformed and normalised into balancing weights. Training across multiple unlabelled observational datasets amortises this computation, allowing the trained model to estimate ATEs on unseen datasets without per-dataset retraining.

FiP and CInA thus use transformers for different causal tasks. FiP learns a causal generative model after inferring an ordering, whereas CInA learns to produce balancing weights for treatment-effect estimation. Both illustrate how computation can be amortised across datasets, but neither removes the structural or identification assumptions required for causal conclusions.

Other foundation models with causal considerations. Beyond these two methods, the interplay between causality and foundation models is being explored from several aspects, and we highlight two active groups of directions.

The first treats causal reasoning as one of the important general reasoning capabilities [393, 1793] of foundation models. The question is whether large foundation models can perform classical causal tasks, such as causal discovery and causal inference, from their pretrained knowledge or in context. A growing body of evaluations probes this ability [846, 885], ranging from causal-discovery benchmarks in different domains [998, 1641] to broader critiques arguing that such models often recite memorised causal facts rather than reason causally [1836, 1841]. These efforts sit alongside related evaluations of tabular-data and graph reasoning [1642], and are increasingly viewed as part of assessing a foundation model’s reasoning capabilities more generally.

The second angle concerns *world models*, which aim to go beyond pattern recognition by capturing how actions lead to consequences. The specific learning methods vary widely, and the tasks and evaluations can only approximate the true underlying dynamics of the world. Causal structure is especially important in two settings: agentic foundation models and scientific foundation models. In *physical intelligence*, where models are used to control embodied agents, correctly modelling the effect of actions becomes essential [690, 1734]; a similar need for causal, verifiable state transitions arises for long-horizon digital agents [633, 1778]. In the sciences, for example drug discovery and biological design, the value of a world model lies in predicting the outcome of interventions such as candidate treatments [349, 1420, 1598], where causal validity directly determines its usefulness of treatment design.

Chapter 55

Credal BDL and Imprecise Probability in BDL

By now, the reader should be convinced of the predictive uncertainty quantification capabilities of Bayesian Neural Networks (BNNs). While that is true, classical Bayesian Deep Learning (BDL) is not able to distinguish properly between different *sources* of predictive uncertainty. In this chapter, we present Credal Bayesian Deep Learning (CBDL) with the aim to overcome such a shortcoming [236].

Intuitively, CBDL allows us to train an (uncountably) infinite ensemble of BNNs (explored in Part IV of this book), using only finitely many elements. This is possible thanks to prior and likelihood Finitely Generated Credal Sets (FGCSs), a concept from the Imprecise Probability literature. Intuitively, convex combinations of a finite collection of prior-likelihood pairs are able to represent infinitely many such pairs. After training, CBDL outputs a set of posteriors on the parameters of the neural network. At inference time, such posterior set is used to derive a set of (posterior) predictive distributions that is in turn utilised to distinguish between (predictive) Aleatoric and Epistemic Uncertainties, and to quantify them. The predictive set also produces either (i) a collection of outputs enjoying desirable probabilistic guarantees, or (ii) the single output that is deemed the best in the “worst case scenario”, that is, the one having the highest predictive lower probability – another Imprecise-Probabilistic concept. CBDL is more robust than single BNNs to prior and likelihood misspecification, and to distribution shift. Furthermore, it is better at quantifying and disentangling different types of (predictive) uncertainties than single BNNs and ensemble of BNNs.

55.1 Introduction

As we have seen in previous chapters, the Bayesian paradigm provides a rigorous framework to analyse and train uncertainty-aware neural networks, and more generally to support the development of learning algorithms [854]. In addition, it overcomes some of the drawbacks of other deep learning models, namely that they are prone to overfitting, which adversely affects their generalisation capabilities, and that they tend to be overconfident about their predictions when they provide a confidence interval. BNNs, though, are trained using a single prior, which may still suffer from miscalibration and robustness issues [1013].

In this chapter, we introduce Credal Bayesian Deep Learning (CBDL) [236], a procedure that draws on concepts from the Imprecise Probability (IP) literature [62, 235, 237, 1636, 1699], and that can be used for both classification and regression tasks. It captures the ambiguity that the designer faces when selecting which prior to choose for the parameters of a neural network, and which likelihood distribution to choose for the training data at hand.

CBDL can be thought of as a neural network (NN) trained using prior and likelihood Finitely Generated Credal Sets (FGCSs), $\mathcal{P}_{\text{prior}}$ and $\mathcal{P}_{\text{lik}}$, respectively.¹ They are convex sets of probability measures having finitely many extreme elements (the elements that cannot be written as a convex combination of one another), $\text{ex}\mathcal{P}_{\text{prior}}$ and $\text{ex}\mathcal{P}_{\text{lik}}$, respectively (see also Remark 55.2). A very simple example of a prior FGCS $\mathcal{P}_{\text{prior}}$ is the collection of all the convex combinations of two one-dimensional Normal distributions $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, i.e.,

$$\mathcal{P}_{\text{prior}} = \{P : p = \beta\mathcal{N}(\mu_1, \sigma_1^2) + (1 - \beta)\mathcal{N}(\mu_2, \sigma_2^2), \text{ for some } \beta \in [0, 1]\}, \quad (55.1)$$

where we denoted by p the probability density function (pdf) of $P \in \mathcal{P}_{\text{prior}}$. Consequently, in this toy example we

¹Intuitively, the larger these credal sets, the higher prior and likelihood ambiguity the user faces.

have that $\text{ex}\mathcal{P}_{\text{prior}} = \{\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)\}$.² FGCSs are further examined in Section 55.2.2.

Given the use of finitely generated credal sets, CBDL can also be seen as a non-condensed (uncountably) infinite ensemble of BNNs – each BNN corresponding to a pair (P, L) of prior P from the prior FGCS $\mathcal{P}_{\text{prior}}$ and likelihood L from the likelihood FGCS $\mathcal{P}_{\text{lik}}$. Such infinite ensemble, though, is carried out using only finitely many elements, that is, only pairs $(P^{\text{ex}}, L^{\text{ex}})$ of components of the sets of extreme elements $\text{ex}\mathcal{P}_{\text{prior}}$ and $\text{ex}\mathcal{P}_{\text{lik}}$. This is because every element P of $\mathcal{P}_{\text{prior}}$ can be obtained from a convex combinations of the elements of $\text{ex}\mathcal{P}_{\text{prior}}$, and similarly for the likelihood FGCS. After training, CBDL produces a posterior FGCS $\mathcal{P}_{\text{post}}$ on the parameters of the neural network. At inference time, $\mathcal{P}_{\text{post}}$ is used to derive an FGCS $\mathcal{P}_{\text{pred}}$ of posterior predictive distributions; that is, given a new input, a set of plausible distributions over the space of outputs.³ In turn, $\mathcal{P}_{\text{pred}}$ generates a set of outputs – or a single output, depending on the user’s needs – that enjoys desirable probabilistic guarantees, even when the elements of credal set $\mathcal{P}_{\text{pred}}$ are approximated, e.g. using variational inference. CBDL also gives a way of quantifying and disentangling different types of uncertainties within $\mathcal{P}_{\text{pred}}$.

A credal set approach is used to overcome some of the drawbacks of single BNNs. In particular, CBDL allows to counter the criticism to the practice in (standard) Bayesian statistics of (i) using a single, arbitrary prior to represent the initial state of ignorance of the agent, (ii) using non-informative priors to model ignorance, and (iii) using a single, arbitrary likelihood to represent the agent’s knowledge about the sampling model.⁴ As a consequence, credal sets make the analysis more robust to prior and likelihood misspecification. In addition, they make it possible to quantify and distinguish between epistemic and aleatoric uncertainties (EU and AU, respectively). This is desirable in light of several areas of recent Machine Learning (ML) research, such as Bayesian deep learning [415, 879], adversarial example detection [1526], and data augmentation in Bayesian classification [870].

AU refers to the uncertainty that is inherent to the data generating process; as such, it is irreducible. Think, for example, of a coin toss. No matter how many times the coin is tossed, the stochastic variability of the experiment cannot be eliminated. EU, instead, refers to the lack of knowledge about the data generating process; as such, it is reducible. It can be lessened on the basis of additional data. For example, after only a few tosses, we are unable to gauge whether a coin is biased or not, but if we repeat the experiment long enough, this type of uncertainty vanishes. We note in passing that predictive EU cannot be adequately captured using a single BNN [512, 804]. One reason for this is that selecting a unique prior and a unique likelihood implicitly assumes perfect knowledge around the true prior and the true data generating process. Another, more subtle one, is explored in Section 55.2.1. Methods that disentangle between the two types of (predictive) uncertainties that are based on a single distribution are ad-hoc, but are not theoretically well-justified. EU can be typically reduced by retraining the model using an augmented training set [1043] (e.g. via semantic preserving transformations [876], Puzzle Mix [888], etc.) On the other hand, since AU is irreducible, there is an increasing need for ML techniques that are able to detect and flag its excess, so that the user can “proceed with caution”.

Remark 55.1. *EU should not be confused with the concept of epistemic probability [384, 385, 1699]. In the subjective probability literature, epistemic probability can be captured by a single distribution. Its best definition can be found in Walley [1699, Sections 1.3.2 and 2.11.2]. There, the author specifies how epistemic probabilities model logical or psychological degrees of partial belief of the agent. We remark, though, how de Finetti and Walley work with finitely additive probabilities, while in this chapter we use countably additive probabilities.*

The motivation for working with credal sets is threefold: (i) it allows to be robust against prior and likelihood misspecification; (ii) unlike when using a single probability distribution, it permits to represent ignorance in the sense of lack of knowledge; (iii) it allows to quantify and disentangle between EU and AU. A more in-depth discussion can be found in Caprio et al. [236, Appendix A]. In addition, we point out that despite a hierarchical Bayesian model (HBM) approach seems to be a viable alternative to one based on credal sets, these latter are better justified philosophically, and do not suffer from the same theoretical shortcomings of HBM procedures [130, 804, 840, 1699]. A more detailed explanation can be found in Caprio et al. [236, Appendix B]. Let us mention that, in Section 55.4, CBDL is only related with Bayesian techniques because a comparison with non-Bayesian ones is either difficult to justify, or unfair. Indeed, the initial knowledge component, captured by the prior distribution (or the set of priors, in our case), has no direct comparisons with other approaches. Take conformal prediction [94, 625, 1498, 1689] for instance. It is a model-free method, which means that no prior knowledge around the experiment at hand is required. The only choice the user makes is which non-conformity score to

²To be more precise, the elements of $\text{ex}\mathcal{P}_{\text{prior}}$ are the two *probability measures* whose pdf’s are the two Normals $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$.

³As we shall see in Section 55.3.1, since computing the posteriors and the predictive distributions is oftentimes an intractable problem, the elements of $\mathcal{P}_{\text{post}}$ and of $\mathcal{P}_{\text{pred}}$ are approximated using Variational Inference (VI, studied in detail in Part III of this book). As a consequence, we denote the VI-approximated posterior and predictive credal sets as $\tilde{\mathcal{P}}_{\text{post}}$ and $\tilde{\mathcal{P}}_{\text{pred}}$, respectively.

⁴Criticisms (i) and (iii) are also pointed out in Manchingal and Cuzzolin [1127, Section 2.2].

use. How would this approach be compared with the choice of different priors, which conveys the idea that the user knows that a true distribution over the space of neural network parameters exists, but it is not fully known? In addition, despite the ubiquitous claim that conformal prediction (CP) is an uncertainty *quantification* tool, it is actually not. CP is an uncertainty *representation* tool. Indeed, CP *represents* uncertainty via the conformal prediction region (on the other hand, in this chapter, uncertainty is represented via credal sets). It does not quantify it: there is no real value attached to any kind of predictive uncertainty (aleatoric or epistemic). Some claim that the size of the conformal prediction region quantifies the uncertainty, but even in that case, it is unable to distinguish between AU and EU. Indeed, the size is a positive function of both: it increases as both increase, and hence it cannot be used to distinguish between the two.

We summarise the content of this chapter next: (1) We present CBDL, and develop the theoretical tools and the algorithm required to use it in practice. (2) We show that a CBDL approach is more robust than single BNNs to prior and likelihood misspecification, and to distribution shifts. We also explain how, during inference, the credal set of posteriors $\mathcal{P}_{\text{post}}$ on the network parameters obtained during training is used to derive a credal set of predictive distributions $\mathcal{P}_{\text{pred}}$, and in turn a set of outcomes – or a single outcome – that enjoys probabilistic guarantees.⁵ (3) We show how CBDL is better at quantifying and disentangling predictive AU and EU than single BNNs and ensemble of BNNs.⁶

Before moving on, let us add two comments. First, the proofs for the propositions in this chapter can be found in Caprio et al. [236, Appendix M], and the downstream tasks capabilities of CBDL are demonstrated in Caprio et al. [236, Section 4.2], where CBDL is applied to two safety-critical systems (motion prediction for autonomous driving and human insulin and blood glucose dynamics for artificial pancreas control). Second, while CBDL pays a computational price coming from the use of credal sets, it is able to quantify both (predictive) EU and AU, and to do so in a principled manner, unlike single and ensembles of BNNs. In addition, it requires less stringent assumptions on the nature of the prior and likelihood ambiguity faced by the agent than other imprecise-probabilities-based techniques, as explained in Caprio et al. [236, Appendices B and L]. The existing work related to the ideas introduced in the present chapter can be found in Caprio et al. [236, Section 5].

Structure of the Chapter. Section 55.2 presents the needed preliminary concepts, followed by Section 55.3 that introduces and discusses the CBDL algorithm, together with its theoretical properties. The experimental results are presented in Section 55.4. Section 55.5 concludes this chapter.

Notation Summary. In this chapter, we make extensive use of different notation and acronyms. We summarise them in Table 55.1 to help the reader navigate through them.

55.2 Background and Preliminaries

In this Section, we present the background notions that are needed to understand our main results. In Section 55.2.1 we argue why posterior predictive distributions are not able to gauge predictive Epistemic Uncertainty. Section 55.2.2 discusses (finitely generated) credal sets, upper and lower probabilities, and imprecise highest density regions. Section 55.2.3 introduces the concepts of upper and lower entropy, which are used to quantify and disentangle AU and EU. The reader familiar with these concepts can skip to Section 55.3.

55.2.1 Posterior Predictive Distributions and EU

Suppose that we collect a new (test) input x_* of interest, and that we want to predict its correct output y_* . In this case, the posterior distribution $p(\theta \mid \mathcal{D})$ over the network parameters comes to the rescue, as it allows us to compute the so-called *posterior predictive distribution*,

$$p(y_* \mid x_*, \mathcal{D}) = \int_{\Theta} p(y_* \mid x_*, \theta) p(\theta \mid \mathcal{D}) d\theta = \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D})} [p(y_* \mid x_*, \theta)], \quad (55.2)$$

where $p(y_* \mid x_*, \theta)$ is the model distribution (i.e. the likelihood that we choose). Posterior predictive $p(y_* \mid x_*, \mathcal{D})$ tells us “how likely” output y_* is to be the “correct one” for input x_* , given the knowledge encapsulated in the training data $\mathcal{D}$, which enters the computation via the posterior probability $p(\theta \mid \mathcal{D})$. Oftentimes, scholars claim

⁵Once again, as we shall see in Section 55.3.1, the elements of the posterior and the predictive credal sets are approximated using Variational Inference (VI).

⁶We note in passing how recently Mucsányi et al. [1204] show that (finite) deep ensembles are the current state-of-the-art methods for quantifying and disentangling different types of uncertainties on ImageNet. The experimental results that we present show that CBDL improves on (finite) ensembles of BNNs, both in uncertainty quantification and disentanglement, and in downstream tasks performances. Together with the findings in Mucsányi et al. [1204], this is an additional argument in favour of the effectiveness of CBDL.

Notation and Acronyms	Meaning
FGCS	Finitely Generated Credal Set
CBDL	Credal Bayesian Deep Learning
Upper case Latin letters, such as P and L	Probability measures, e.g. prior or likelihood
Lower case Latin letters, such as p and l	The pdf/pmf associated to probability measures
Π	A finite collection of probability measures
$\text{Conv}(\cdot)$	The convex hull operator
Π'	The convex hull of Π , i.e. $\Pi' = \text{Conv}(\Pi)$
$\mathcal{P}_{\text{prior}}$	A prior FGCS
$\mathcal{P}_{\text{lik}}$	A likelihood FGCS
$\mathcal{P}_{\text{post}}$	A posterior FGCS
VI	Variational Inference
$\tilde{\mathcal{P}}_{\text{post}}$	Posterior FGCS with VI-approximated extrema
$\mathcal{P}_{\text{pred}}$	A predictive FGCS
$\hat{\mathcal{P}}_{\text{pred}}$	Posterior predictive FGCS with VI-approximated extrema
$\text{ex}\mathcal{P}$	The extreme elements of an FGCS $\mathcal{P}$
BNN	Bayesian Neural Network
$\mathcal{D}$	Training dataset
EBNN	Ensemble of Bayesian Neural Networks
$x \mapsto f(x)$	Function representing the neural network architecture
$\underline{P}, \bar{P}$	Lower and upper probabilities, respectively
HDR	Highest Density Region
IHDR	Imprecise Highest Density Region
$IR_{\alpha}(\mathcal{P})$	IHDR of level α associated with FGCS $\mathcal{P}$
$H(P)$	Entropy of probability measure P
$\underline{H}(P), \bar{H}(P)$	Upper and lower entropy associated with a credal set
$\text{EU}(\mathcal{P})$	Epistemic Uncertainty encoded in an FGCS $\mathcal{P}$
$\text{AU}(\mathcal{P})$	Aleatoric Uncertainty encoded in an FGCS $\mathcal{P}$
$\text{TU}(\mathcal{P})$	Total Uncertainty encoded in an FGCS $\mathcal{P}$
$\#A$	Cardinality of a generic set A
post, pred	The act of computing posterior and posterior predictive, resp.
BMA	Bayesian Model Averaging

Table 55.1: Notation and acronyms that we use throughout the chapter, together with their meaning.

that, in a Bayesian setting, the distribution on the parameters θ captures the epistemic uncertainty (EU) faced by the agent. This is a somehow agreeable premise, akin to a second-order distribution reasoning. However, if we accept this assertion, we see that EU at the predictive level is no longer quantifiable, since it is washed away by taking the expectation $\mathbb{E}_{\theta \sim p(\theta|\mathcal{D})}[\cdot]$. As we shall see later, CBDL allows us to overcome this conceptual shortcoming.

55.2.2 Imprecise Probabilities

As CBDL is rooted in the theory of imprecise probabilities (IPs), in this Section we give a gentle introduction to the IP concepts we will use throughout the chapter.

CBDL is based on the *Bayesian sensitivity analysis* (BSA) approach to IPs, that in turn is grounded in the *dogma of ideal precision* (DIP) [122], [1699, Section 5.9]. The DIP posits that in any problem there is an *ideal probability model* which is precise, but which may not be precisely known. We call this condition *ambiguity* [488, 626].

Facing ambiguity can be represented mathematically by a set $\mathcal{P}_{\text{prior}}$ of priors and a set $\mathcal{P}_{\text{lik}}$ of likelihoods that seem “plausible” or “fit” to express the agent’s beliefs on the parameters of interest and their knowledge of the data generating process (DGP). Generally speaking, the farther apart the “boundary elements” of the sets (i.e., their infimum and supremum), the higher the agent’s ambiguity. Of course, if $\mathcal{P}_{\text{prior}}$ and $\mathcal{P}_{\text{lik}}$ are singletons we go back to the usual Bayesian paradigm.

A procedure based on sets $\mathcal{P}_{\text{prior}}$ and $\mathcal{P}_{\text{lik}}$ yields results that are more robust to prior and likelihood misspecification than a regular Bayesian method. In the presence of prior ignorance and indecisiveness about the sampling model, it is better to give answers in the form of intervals or sets, rather than arbitrarily select a prior and a likelihood, and then update. Sets $\mathcal{P}_{\text{prior}}$ and $\mathcal{P}_{\text{lik}}$ allow to represent *indecision*, thus leading to less informative but more robust and valid conclusions.

Remark 55.2. Throughout the chapter, we denote by $\Pi = \{P_1, \dots, P_k\}$, $k \in \mathbb{N}$, a finite set of probabilities on a generic space Ξ , such that for all $j \in \{1, \dots, k\}$, P_j cannot be written as a convex combination of the other $k - 1$ components of Π . We denote by Π' its convex hull $\Pi' \equiv \text{Conv}(\Pi)$, i.e., the set of probabilities Q on Ξ that can be written as $Q(A) = \sum_{j=1}^k \beta_j P_j(A)$, for all $A \subset \Xi$, where the β_j ’s are elements of $[0, 1]$ that sum up to 1. In the literature, it is referred to as a *Finitely Generated Credal Set* [FGCS 341, 1017]. Notice then that the extreme elements of Π' correspond to the elements of Π , in formulas $\text{ex}\Pi' = \Pi$. Simple graphical representations of finitely generated credal sets are given in Figures 55.1 and 55.2.

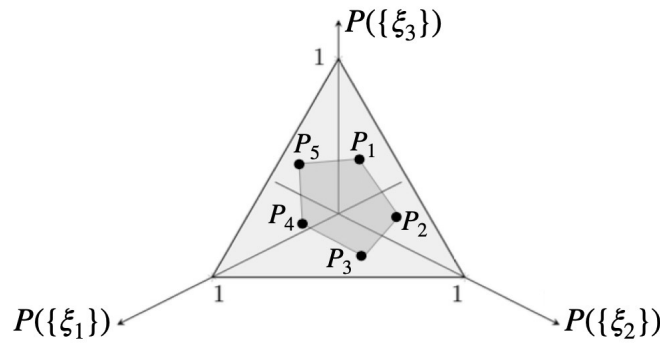


Figure 55.1: Suppose we are in a 3-class classification setting, so $\Xi = \{\xi_1, \xi_2, \xi_3\}$. Then, any probability measure P on Ξ can be seen as a probability vector. For example, suppose $P(\{\xi_1\}) = 0.6$, $P(\{\xi_2\}) = 0.3$, and $P(\{\xi_3\}) = 0.1$. We have that $P \equiv (0.6, 0.3, 0.1)^\top$. Since its elements are positive and sum up to 1, probability vector P belongs to the unit simplex, the triangle in the figure. Then, we can specify $\Pi = \{P_1, \dots, P_5\}$, and obtain as a consequence that $\Pi' = \text{Conv}(\Pi)$ is the pentagon. It is a convex polygon with finitely many extreme elements, and it is the geometric representation of a finitely generated credal set.

Let us now introduce the concepts of *lower* and *upper probabilities*. The lower probability $\underline{P}$ associated with Π is given by $\underline{P}(A) = \inf_{P \in \Pi} P(A)$, for all $A \subset \Xi$. The upper probability $\bar{P}$ associated with Π is defined as the conjugate to $\underline{P}$, that is, $\bar{P}(A) := 1 - \underline{P}(A^c) = \sup_{P \in \Pi} P(A)$,⁷ for all $A \subset \Xi$. These definitions hold even if Π is not finite. Then, we have the following important result.

⁷Here, $A^c := \Xi \setminus A$.

Proposition 55.1. $\bar{P}$ is the upper probability for Π if and only if it is also the upper probability for Π' . That is, $\bar{P}(A) = \sup_{P \in \Pi} P(A) = \sup_{P' \in \Pi'} P'(A)$, for all $A \subset \Xi$. The same holds for the lower probability.

A version of Proposition 55.1 was proven in Dantzig [371], while a variant for finitely additive probability measures can be found in Walley [1699, Section 3.6]. A simple graphical representation of upper and lower probabilities for a set A is given in Figure 55.2.

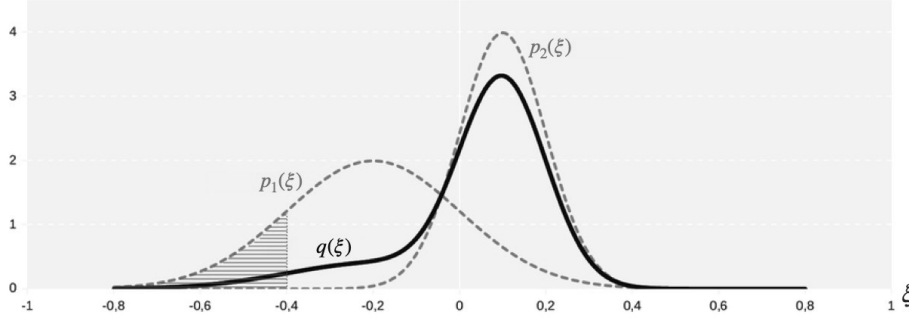


Figure 55.2: In this figure, a replica of Flint and Maré [535, Figure 1], $\Pi = \{P_1, P_2\}$, where P_1 and P_2 are two Normal distributions whose probability density functions (pdf's) p_1 and p_2 are given by the dashed curves. Their convex hull is $\Pi' = \text{Conv}(\Pi) = \{Q : Q = \beta P_1 + (1 - \beta)P_2, \text{ for all } \beta \in [0, 1]\}$. The pdf q of an element Q of Π' is depicted by a solid black curve. In addition, let $A = [-0.8, -0.4]$. Then, $\underline{P}(A) = \int_{-0.8}^{-0.4} p_2(\xi) d\xi \approx 0$, while $\bar{P}(A)$ is given by the shaded area under p_1 , that is, $\bar{P}(A) = \int_{-0.8}^{-0.4} p_1(\xi) d\xi$.

We now use lower probability $\underline{P}$ to define the α -level *Imprecise Highest Density Region* (IHDR), for some $\alpha \in [0, 1]$.

Definition 55.1. [329, Section 2] Let α be any value in $[0, 1]$. Then, set $IR_\alpha(\Pi') \subset \Xi$ is called a $(1 - \alpha)$ -Imprecise Highest Density Region (IHDR) if

1. $\underline{P}[\{\xi \in IR_\alpha(\Pi')\}] \geq 1 - \alpha$;
2. $\int_{IR_\alpha(\Pi')} d\xi$ is minimal. If Ξ is at most countable, we replace $\int_{IR_\alpha(\Pi')} d\xi$ with $\#IR_\alpha(\Pi')$, where $\#$ denotes the cardinality operator.

Definition 55.1 holds also if $\Pi = \text{ex}\Pi'$ is not finite. Notice that condition 2 is needed so that $IR_\alpha(\Pi')$ is the subset of Ξ having the lowest possible cardinality, which still satisfies condition 1. By the definition of lower probability, Definition 55.1 implies that $P'[\{\xi \in IR_\alpha(\Pi')\}] \geq 1 - \alpha$, for all $P' \in \Pi'$. Here lies the appeal of the IHDR concept. Let us give a simple example, borrowed from Caprio et al. [239]. Suppose $\Xi = \{\xi_1, \dots, \xi_5\}$, $\Pi = \{P_1, P_2, P_3\}$, and $\alpha = 0.1$. The numerical values for $P_s(\{\xi_j\})$ are given in Table 55.2, for all $s \in \{1, 2, 3\}$ and all $j \in \{1, \dots, 5\}$. Then, from Proposition 55.1 and Definition 55.1, we have that $IR_\alpha(\Pi') = \{\xi_1, \xi_2, \xi_3\}$.

	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5
P_1	0.7	0.25	0.03	0.01	0.01
P_2	0.6	0.2	0.1	0.05	0.05
P_3	0.5	0.3	0.15	0.025	0.025

Table 55.2: Numerical values for our example. It is easy to see that the smallest subset of Ξ that is assigned a probability of at least **0.9** by all the elements of Π is $\{\xi_1, \xi_2, \xi_3\}$.

An operative way of building the IHDR is to consider the union of the (precise) Highest Density Regions (HDRs) of the elements of $\Pi = \text{ex}\Pi'$.⁸ Let us define them formally.

Definition 55.2. [329, Section 1] Pick any $P_j \in \Pi$, $j \in \{1, \dots, k\}$. Let α be any value in $[0, 1]$. Then, set $R_\alpha(P_j) \subset \Xi$ is called a $(1 - \alpha)$ -Highest Density Region (HDR) for P_j if

$$P_j[\{\xi \in R_\alpha(P_j)\}] \geq 1 - \alpha \quad \text{and} \quad \int_{R_\alpha(P_j)} d\xi \text{ is minimal.} \quad (55.3)$$

⁸Here, by “operative” we mean that this procedure to compute the IHDR is easy to carry out in practice.

If Ξ is at most countable, we replace $\int_{R_\alpha(P_j)} d\xi$ with $\#R_\alpha(P_j)$. Equivalently [806],

$$R_\alpha(P_j) = \{\xi \in \Xi : p_j(\xi) \geq p_j^\alpha\} \subset \Xi, \quad (55.4)$$

where p_j is the pdf or the probability mass function (pmf) of P_j , and p_j^α is a constant value. In particular, it is the largest constant such that $P_j[\{\xi \in R_\alpha(P_j)\}] \geq 1 - \alpha$.

In dimension 1, $R_\alpha(P_j)$ can be interpreted as the smallest collection of elements of Ξ (interval or union of intervals) to which distribution P_j assigns probability of at least $1 - \alpha$. As we can see, HDRs are a Bayesian counterpart of confidence intervals.⁹ We give a simple visual example in Figure 55.3.

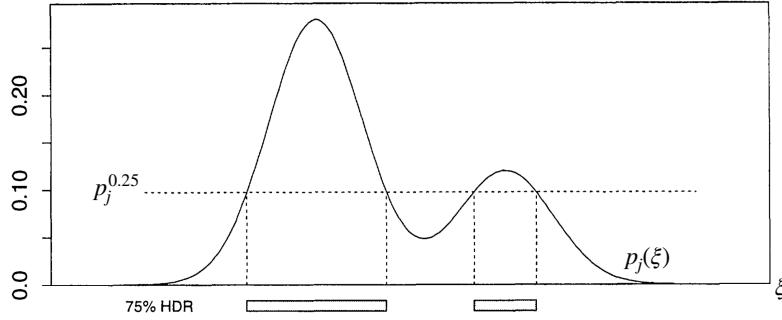


Figure 55.3: The 0.25-HDR from a Normal Mixture density. This picture is a replica of Hyndman [806, Figure 1]. The geometric representation of “75% probability according to P_j ” is the area between the ξ axis and the part of the density p_j where $p_j(\xi) \geq p_j^{0.25}$ is equal to $1 - 0.25 = 0.75$.

As we mentioned earlier, an operative way of obtaining IHDR $IR_\alpha(\Pi')$ is by putting $IR_\alpha(\Pi') = \cup_{j=1}^k R_\alpha(P_j)$. Thanks to Proposition 55.1, by taking the union of the HDRs, we ensure that all the probability measures in the credal set $\Pi' = \text{Conv}(\Pi)$ assign probability of at least $1 - \alpha$ to the event $\{\xi \in IR_\alpha(\Pi')\}$. In turn, this implies that $\underline{P}[\{\xi \in IR_\alpha(\Pi')\}] = \min_{j \in \{1, \dots, k\}} P_j[\{\xi \in IR_\alpha(\Pi')\}] \geq 1 - \alpha$. We also have that the difference between the upper and lower probabilities of $\{\xi \in IR_\alpha(\Pi')\}$ is bounded by α . To see this, notice that $\bar{P}[\{\xi \in IR_\alpha(\Pi')\}] \leq 1$, so $\bar{P}[\{\xi \in IR_\alpha(\Pi')\}] - \underline{P}[\{\xi \in IR_\alpha(\Pi')\}] \leq \alpha$.

55.2.3 Quantifying and Disentangling Aleatoric and Epistemic Uncertainties

Recall that, given a probability measure P on a generic space Ξ , the (Shannon) entropy of P is defined as $H(P) := \mathbb{E}[-\log p] = -\int_{\Xi} \log[p(\xi)]P(d\xi)$ if Ξ is uncountable, where p denotes the pdf of P . If Ξ is at most countable, we have that $H(P) = -\sum_{\xi \in \Xi} P(\{\xi\}) \log[P(\{\xi\})]$. As pointed out by Dubois and Hüllermeier [457], Hüllermeier and Waegeman [804], the entropy primarily captures the shape of the distribution, namely its “peakedness” or non-uniformity, and hence informs about the predictability of the outcome of a random experiment: the higher its value, the lower the predictability. Then, we can define the imprecise versions of the Shannon entropy as proposed by Abellán et al. [4], Hüllermeier and Waegeman [804], $\bar{H}(P) := \sup_{P \in \Pi'} H(P)$ and $\underline{H}(P) := \inf_{P \in \Pi'} H(P)$, called the *upper* and *lower Shannon entropy*, respectively. Notice that these definitions hold for all sets of probabilities, not just for (finitely generated) credal sets. The upper entropy is a measure of total uncertainty since it represents the minimum level of predictability associated with the elements of Π' . In Abellán et al. [4], Hüllermeier and Waegeman [804], the authors posit that it can be decomposed as a sum of aleatoric and epistemic uncertainties, and that the latter can be specified as the difference between upper and lower entropy, thus obtaining

$$\underbrace{\bar{H}(P')}_{\text{TU}(\Pi')} = \underbrace{\underline{H}(P')}_{\text{AU}(\Pi')} + \underbrace{[\bar{H}(P') - \underline{H}(P')]}_{\text{EU}(\Pi')}, \quad (55.5)$$

where $\text{TU}(\Pi')$ denotes the total uncertainty associated with set Π' , $\text{AU}(\Pi')$ is the AU associated with Π' , and $\text{EU}(\Pi')$ represents the EU associated with Π' . As we can see, if Π' is a singleton, then $\text{TU}(\Pi') = \text{AU}(\Pi')$, and $\text{EU}(\Pi') = 0$. This captures the idea that, in general, a single distribution can only gauge aleatoric uncertainty.¹⁰ We have the following proposition.

⁹Hence standard choices for the value of α are 0.1, 0.05, 0.01.

¹⁰It can gauge EU, though, if it is a second-order distribution, or if it is the result of an ensemble of probabilities.

Algorithm 39: Credal Bayesian Deep Learning (CBDL) – Training and Inference*During Training***Step 1** Specify K priors $\text{ex}\mathcal{P}_{\text{prior}} = \{P_k^{\text{ex}}\}_{k=1}^K$ **Step 2** Specify S likelihoods $\text{ex}\mathcal{P}_{\text{lik}} = \{L_s^{\text{ex}}\}_{s=1}^S$ **Step 3** Compute $P_{k,s}(\cdot | \mathcal{D}) = \text{post}[P_k^{\text{ex}}, L_s^{\text{ex}}]$, for all k and s `{% VI-Appr. by  $\hat{P}_{k,s}$  %}`*During Inference***Inputs** : New input $x_* \in \mathcal{X}$ **Params** : Confidence parameter $\alpha \in [0, 1]$ **Outputs:** Predictive Aleatoric and Epistemic Uncertainties, α -level IHDR**Step 4** Compute $P_{k,s}^{\text{pred}} = \text{pred}[\hat{P}_{k,s}, L_s^{\text{ex}}]$, for all k and s `{% VI-Appr. by  $\hat{P}_{k,s}^{\text{pred}}$  %}`**Step 5** Compute and return $\text{AU}(\hat{\mathcal{P}}_{\text{pred}})$ and the bounds for $\text{EU}(\hat{\mathcal{P}}_{\text{pred}})$ **Step 6** Compute and return the $(1 - \alpha)$ -IHDR $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$

Proposition 55.2. *Let Π, Π' be sets of probability measures as the ones considered in Remark 55.2. Then, $\sup_{P \in \Pi} H(P) = \overline{H}(P) \leq \overline{H}(P') = \sup_{P' \in \Pi'} H(P')$ and $\inf_{P \in \Pi} H(P) = \underline{H}(P) = \underline{H}(P') = \inf_{P' \in \Pi'} H(P')$.*

Proposition 55.2 tells us that the upper entropy of the extreme elements in $\Pi = \text{ex}\Pi'$ is a lower bound for the upper entropy of the whole credal set Π' , and that the lower entropy of the extreme elements in Π is equivalent to the lower entropy of the whole credal set Π' . These facts imply that $\text{AU}(\Pi') = \text{AU}(\Pi)$, and that $\text{EU}(\Pi') \geq \text{EU}(\Pi)$. In addition, as a consequence of Smieja and Tabor [1523, Theorem III.1], we have that $\text{EU}(\Pi') \leq \text{EU}(\Pi) + \log(\#\Pi)$. In turn, we have that $\text{EU}(\Pi') \in [\text{EU}(\Pi), \text{EU}(\Pi) + \log(\#\Pi)]$.

We briefly mention that other uncertainty measures based on credal sets are also available (see Bronevich and Rozenberg [209], Caprio et al. [238], Hofman et al. [766], Hüllermeier and Waegeman [804] for a few examples) and they can be used in place of upper and lower entropy¹¹ to quantify EU and AU within the credal set $\mathcal{P}$, as long as the measure chosen for the total uncertainty is bounded.

Let us also add a small remark. Although EU can be reduced with an increasing amount of data, it is very seldom the case that it goes to zero when a finite amount of data is collected. When that happens, it means that the initial uncertainty was very low in the first place. Typically, EU goes to zero only asymptotically, as a finite amount of data is almost never enough to overcome initial uncertainty [1699, 1773].

55.3 CBDL and Its Properties

This is the main portion of the chapter. In Section 55.3.1, we present and discuss the CBDL algorithm. Its theoretical properties are derived in Section 55.3.2.

55.3.1 CBDL algorithm

Recall that $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denotes the training set, and let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output spaces, respectively. We then denote by P a generic prior on the parameters $\theta \in \Theta$ of a BNN having pdf p , and by $L \equiv L_{x,\theta}$ a generic likelihood on the space $\mathcal{Y}$ of outputs having pdf $l \equiv l_{x,\theta}$. The act of computing the posterior from prior P and likelihood L using a BNN is designated by $\text{post}[P, L]$, and the act of deriving the posterior predictive from posterior $P(\cdot | \mathcal{D})$ and likelihood L is designated by $\text{pred}[P(\cdot | \mathcal{D}), L]$. The CBDL procedure is presented in Algorithm 39, and it is discussed in the following paragraphs.

During training, in **Step 1** the user specifies K priors on the parameters of the neural network, that constitute the extrema $\text{ex}\mathcal{P}_{\text{prior}}$ of the prior FGCS $\mathcal{P}_{\text{prior}}$. Similarly, in **Step 2** they elicit S likelihoods capturing the possible architectures of the neural network, that correspond to the extrema $\text{ex}\mathcal{P}_{\text{lik}}$ of the likelihood FGCS $\mathcal{P}_{\text{lik}}$. Let us give an example. Outlined in Jospin et al. [854, Sections IV-B and IV-C1], for classification, the standard process for BNNs involves

- A Normal prior with zero mean and diagonal covariance $\sigma^2 I$ on the coefficients of the network, that is, $p(\theta) = \mathcal{N}(0, \sigma^2 I)$. In the context of CBDL, we could specify, e.g. $\text{ex}\mathcal{P}_{\text{prior}} = \{P : p(\theta) = \mathcal{N}(\mu, \sigma^2 I), \mu \in \{\mu_-, \mathbf{0}, \mu_+\}, \sigma^2 \in \{3, 7\}\}$. That is, the extreme elements of the prior credal set are six independent Normals having different levels of “fatness” of the tails, and centred at a vector μ_+ having positive entries, a vector

¹¹Which, although easy to compute, can sometimes suffer from shortcomings coming from lacking the monotonicity property. That is, there are credal sets that, although nested into each other, have the same upper and/or lower entropy [804].

μ_- having negative entries, and a vector $\mathbf{0}$ having entries equal to 0. They capture the ideas of positive bias, negative bias, and no bias of the coefficients, respectively. This is done to hedge against possible prior misspecification.

- A Categorical likelihood, $p(y | x, \theta) = \text{Cat}(f_\theta(x))$, whose parameter is given by the output of a functional model f_θ (e.g. a neural network having parameters θ). In the context of CBDL, we could specify the set of extreme elements of the likelihood credal set as $\text{ex}\mathcal{P}_{\text{lik}} = \{L : l_{x,\theta}(y) = \text{Cat}(f_{s,\theta}(x)), s \in \{1, \dots, S\}\}$. Specifying set $\text{ex}\mathcal{P}_{\text{lik}}$, then, corresponds to eliciting a finite number S of possible (parametrised) architectures for the neural network $f_{s,\theta}$, $s \in \{1, \dots, S\}$, and obtain, as a consequence, S categorical distributions $\{\text{Cat}(f_{s,\theta}(x))\}_{s=1}^S$. This captures the ambiguity around the true data generating process faced by the agent, and allows them to hedge against likelihood misspecification.

More in general, we can use the priors and likelihoods that better fit the type of analysis we are performing.¹² For example, for the choice of the priors we refer to Fortuin et al. [548] and to Part VI of this book, where the problem of selecting the right type of prior for BNNs, and their properties, are studied in depth.

Step 3 performs an element-wise application of Bayes' rule for all the elements of $\text{ex}\mathcal{P}_{\text{prior}}$ and $\text{ex}\mathcal{P}_{\text{lik}}$. Each posterior is approximated using the Variational Inference (VI) method, studied in detail in Part III of this book. That is, we project every posterior $P_{k,s}(\cdot | \mathcal{D})$ onto a set $\mathbb{S}$ of “well-behaved” distributions (e.g. Normals) using the KL divergence. In formulas, $\check{P}_{k,s} = \arg \min_{Q \in \mathbb{S}} KL[Q \| P_{k,s}(\cdot | \mathcal{D})]$. By “well-behaved”, we mean that they have to satisfy the conditions in Zhang and Gao [1844, Sections 2 and 3].¹³ This ensures that as the sample size goes to infinity, the approximated posteriors converge to the true data generating process. We also point out how despite using a VI approximation for the exact posteriors, as we shall see in the next Section, the credal set of approximated posteriors is closer to the “oracle” posterior $P^o(\cdot | \mathcal{D})$ than any of its elements taken individually. As a consequence, working with credal sets leads to VI posterior approximations that are better than the ones resulting from a single BNN, or an ensemble of BNNs, where several BNNs are combined into one.

Remark 55.3. *Although highly unlikely in practice, it is theoretically possible that the VI approximation of the (finite) set $\{P_{k,s}(\cdot | \mathcal{D})\}_{k,s}$ of posteriors is a singleton, see Figure 55.4. While the conditions in Zhang and Gao [1844] guarantee that asymptotically the approximated posteriors coincide with the true data generating process, this typically does not happen with finite-dimensional datasets. As a consequence, obtaining a singleton when projecting $\{P_{k,s}(\cdot | \mathcal{D})\}_{k,s}$ onto $\mathbb{S}$ may result in an underestimation of the uncertainties faced by the user. In that case, we either consider a different set – whose elements still satisfy the conditions in Zhang and Gao [1844, Sections 2 and 3] – on which to project $\{P_{k,s}(\cdot | \mathcal{D})\}_{k,s}$ according to the KL divergence, or we use a different “projection operator”, that is, a divergence different from the KL. For example, Rényi and χ^2 divergences, or Hellinger and total variation metrics are suggested by Zhang and Gao [1844]. Alternatively, we can consider a different approximation strategy altogether, for instance the Laplace approximation (see e.g. Ritter et al. [1402] and Part II of this book).*

After **Step 3**, we obtain a finite set $\{\check{P}_{k,s}\}_{k,s}$ of VI approximation of the posteriors on the network parameters, whose cardinality is $K \times S$. Its convex hull constitutes $\check{\mathcal{P}}_{\text{post}}$, that is, the VI-approximated posterior FGCS. We assume that $\text{ex}\check{\mathcal{P}}_{\text{post}} = \{\check{P}_{k,s}\}_{k,s}$. This is an assumption because it may well be that – due to the approximation procedure – some of the elements of $\{\check{P}_{k,s}\}_{k,s}$ are not independent of one another. An interesting open research question is that of designing a procedure that finds the elements of $\{\check{P}_{k,s}\}_{k,s}$ which cannot be written as a convex combination of one another.

Being a combinatorial task, **Step 3** is a computational bottleneck of Algorithm 39. We have to calculate $K \times S$ VI approximations to as many posteriors, but this allows us to forego any additional assumptions on the nature of the lower and upper probabilities that are oftentimes required by other imprecise-probabilities-based techniques.¹⁴ Clearly, CBDL is simplified if either $\mathcal{P}_{\text{prior}}$ or $\mathcal{P}_{\text{lik}}$ are singletons. Notice that in the case that $\mathcal{P}_{\text{prior}}$ and $\mathcal{P}_{\text{lik}}$ are both not singletons, for all $A \subset \Theta$ the interval $[\check{P}(A), \bar{P}(A)]$ is wider than the case when one or the other is a singleton. In the limiting case where both are singletons, we retrieve the usual Bayesian updating, so the interval shrinks down to a point.

Before moving to inference time, let us remark a difference between Bayesian Model Averaging (BMA) and CBDL. In BMA, the user specifies a distribution on the models. Translated in the notation we use in this chapter, this means having a discrete distribution Q over the $K \times S$ prior-likelihood combinations, that is, over the elements

¹²Choosing 2 to 5 priors and likelihoods is usually enough to safely hedge against prior and likelihood misspecification.

¹³We assume that the conditions on the priors and the likelihoods given in Zhang and Gao [1844] are satisfied.

¹⁴If we are willing to make such assumptions, Caprio et al. [236, Theorem 10] shows how to compute the upper posterior using only upper prior and upper likelihood.

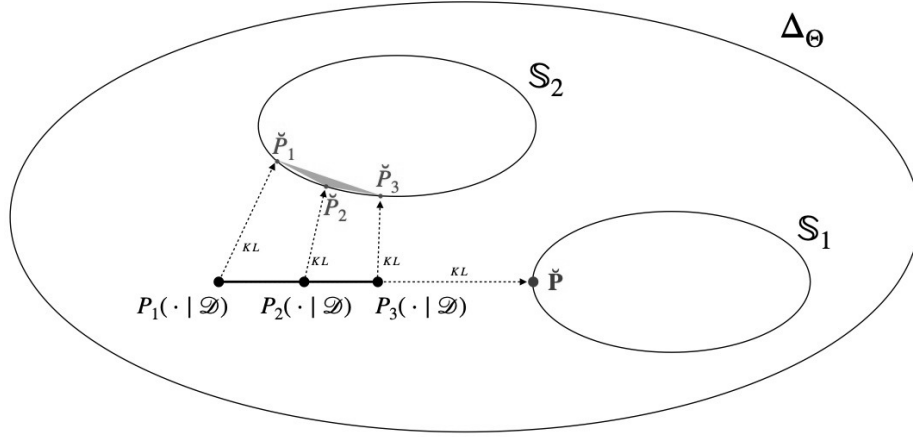


Figure 55.4: Let Δ_Θ denote the space of probability measures on Θ . Suppose that in the analysis at hand we specified three priors and only one likelihood, so $S = 1$ and we can drop the s index. Let $\{P_k(\cdot | \mathcal{D})\}_{k=1}^3$ be the collection of exact posteriors, so that the solid black segment represents the exact posterior FGCS. Then, if we project the elements of $\{P_k(\cdot | \mathcal{D})\}_{k=1}^3$ onto S_1 via the KL divergence, we obtain the same distribution $\check{P}$. This is detrimental to the analysis because such an approximation underestimates the (posterior) epistemic (and possibly also aleatoric) uncertainty faced by the agent. Then, the user could specify a different set S_2 of “well-behaved” distributions onto which project the elements of $\{P_k(\cdot | \mathcal{D})\}_{k=1}^3$. In the figure, we see that they are projected onto S_2 via the KL divergence to obtain $\check{P}_1$, $\check{P}_2$, and $\check{P}_3$. The convex hull of these latter, captured by the shaded triangle, represents the variational approximation of the exact posterior FGCS.

of $\text{ex}\check{\mathcal{P}}_{\text{post}}$. Such Q is then used to select an element $\check{P}^*$ from the (VI-approximated) posterior FGCS $\check{\mathcal{P}}_{\text{post}}$ as

$$\check{\mathcal{P}}_{\text{post}} \ni \check{P}^* = \sum_{k,s} Q(\{\check{P}_{k,s}\}) \check{P}_{k,s}, \quad (55.6)$$

where $Q(\{\check{P}_{k,s}\}) \in [0, 1]$ for all k and all s , and $\sum_{k,s} Q(\{\check{P}_{k,s}\}) = 1$ [234]. For example, if Q is the discrete uniform distribution on $\{1, \dots, K \times S\}$, then $\check{P}^*$ is the so-called *centre of gravity* of the credal set $\check{\mathcal{P}}_{\text{post}}$ [1185, Section 3.2]. Instead, CBDL does not select a unique distribution from $\check{\mathcal{P}}_{\text{post}}$. Rather, distributions are kept separate so to be able to derive a predictive FGCS in **Step 4** of Algorithm 39, that is in turn used to quantify and disentangle predictive uncertainties, and to compute the predictive IHDR, that is, a collection of outputs having a high probability of being the correct ones for a new input x_* .

During inference, a new input x_* is provided. In **Step 4**, every element of $\text{ex}\check{\mathcal{P}}_{\text{post}}$ is used to derive a predictive distribution $P_{k,s}^{\text{pred}} = \text{pred}[\check{P}_{k,s}, L_s^{\text{ex}}]$ on the output space $\mathcal{Y}$. In particular, for every k and every s , the pdf $p_{k,s}^{\text{pred}}$ of $P_{k,s}^{\text{pred}}$ is obtained as

$$\begin{aligned} p_{k,s}^{\text{pred}}(y_* | x_*, x_1, y_1, \dots, x_N, y_N) &= \int_{\Theta} l_s^{\text{ex}}(y_* | x_*, \theta) \cdot p_{k,s}(\theta | x_1, y_1, \dots, x_N, y_N) d\theta \\ &\approx \int_{\Theta} l_s^{\text{ex}}(y_* | x_*, \theta) \cdot \check{p}_{k,s}(\theta) d\theta, \end{aligned} \quad (55.7)$$

where l_s^{ex} is the pdf of likelihood L_s^{ex} , $p_{k,s}$ is the pdf of the true posterior $P_{k,s}(\cdot | \mathcal{D})$, $\check{p}_{k,s}$ is the pdf of the VI-approximated posterior $\check{P}_{k,s}$, and y_* is the output associated to the new input x_* (see Caprio et al. [236, Appendix F] for more details). Each predictive distribution $P_{k,s}^{\text{pred}}$ is approximated to $\hat{P}_{k,s}^{\text{pred}}$ (e.g. using Normals) via Variational Inference. The convex hull of the collection $\{\hat{P}_{k,s}^{\text{pred}}\}_{k,s}$ having cardinality $K \times S$ constitutes the VI-approximated predictive FGCS $\hat{\mathcal{P}}_{\text{pred}}$. Similarly to what we did for $\text{ex}\check{\mathcal{P}}_{\text{post}}$, we assume that $\text{ex}\hat{\mathcal{P}}_{\text{pred}} = \{\hat{P}_{k,s}^{\text{pred}}\}_{k,s}$.

In **Step 5**, building on the results in Section 55.2.3, and in particular on Proposition 55.2, we compute and return $\text{AU}(\hat{\mathcal{P}}_{\text{pred}})$ and the bounds for $\text{EU}(\hat{\mathcal{P}}_{\text{pred}})$. In particular, we have

$$\text{AU}(\mathcal{P}_{\text{pred}}) \approx \text{AU}(\hat{\mathcal{P}}_{\text{pred}}) = \underline{H}(\hat{\mathcal{P}}_{\text{pred}}) \quad (55.8)$$

and

$$\text{EU}(\mathcal{P}_{\text{pred}}) \approx \text{EU}(\hat{\mathcal{P}}_{\text{pred}}) \in [\bar{H}(\hat{\mathcal{P}}_{\text{pred}}) - \underline{H}(\hat{\mathcal{P}}_{\text{pred}}), \bar{H}(\hat{\mathcal{P}}_{\text{pred}}) - \underline{H}(\hat{\mathcal{P}}_{\text{pred}}) + \log(K \times S)], \quad (55.9)$$

where (i) $\underline{H}(\hat{P}^{\text{pred}}) = \min_{k,s} H(\hat{P}_{k,s}^{\text{pred}})$; (ii) $\overline{H}(\hat{P}^{\text{pred}}) = \max_{k,s} H(\hat{P}_{k,s}^{\text{pred}})$; and (iii) $\mathcal{P}_{\text{pred}}$ is the “true” predictive FGCS, that is, the one we would have obtained had we been able to compute the exact posteriors in **Step 3** and the exact predictive distributions in **Step 4**. Let us point out a salient feature of CBDL. The AU and the (bounds for) the EU associated with the VI-approximated predictive FGCS $\hat{\mathcal{P}}_{\text{pred}}$ embed uncertainty comparable to an uncountably infinite ensemble of BNNs, i.e., an ensemble of BNNs of cardinality $\aleph_1$, despite the simple and intuitive mathematics over the finite set $\text{ex}\hat{\mathcal{P}}_{\text{pred}}$. They are not merely pessimistic results on the uncertainties associated with a finite ensemble of BNNs.

We observe that we compute the AU and EU associated with the (VI-approximated) predictive credal set $\hat{\mathcal{P}}_{\text{pred}}$, rather than those related to the (VI-approximated) posterior credal set $\hat{\mathcal{P}}_{\text{post}}$. We do so because we are ultimately interested in reporting the uncertainty around the predicted outputs given a new input x_* in the problem at hand, more than the uncertainty on the parameters of the NN.

Before commenting on the next step, let us pause here and add a remark. While a “bad choice” of priors and likelihoods may lead to large values of upper and lower entropy – $\overline{H}(\hat{P}^{\text{pred}})$ and $\underline{H}(\hat{P}^{\text{pred}})$, respectively – this is not a risk confined to CBDL. Poor modelling choices are an unavoidable risk in model-based techniques. This gave rise to the famous adage by George Box “essentially, all models are wrong but some are useful” [199].¹⁵ We maintain that CBDL is indeed useful, since it overcomes some of the shortcomings of traditional Bayesian techniques – as explained in Caprio et al. [236, Appendices A and B]. As for “regular” Bayesian methods, though, for CBDL too the designer will need to make “plausible” choices for priors and likelihoods. A further positive aspect of working with credal sets is that they are able to “self-regulate”. That is, in the case of prior-likelihood conflict – which happens e.g. if the prior set is ill-specified [1389] – the posterior credal set, and in turn the predictive credal set too, will be wider. This will be reflected in the uncertainty measures which will register an excess of posterior, and in turn predictive, epistemic uncertainties.

Finally, in **Step 6**, we compute and return the α -level Imprecise Highest Density Region $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ for the VI-approximated predictive FGCS $\hat{\mathcal{P}}_{\text{pred}}$, which approximates the IHDR for the “true” predictive FGCS $\mathcal{P}_{\text{pred}}$. It is the smallest subset of $\mathcal{Y}$ such that $\hat{P}^{\text{pred}}[\{y_* \in IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})\}] \geq 1 - \alpha$, for all $\hat{P}^{\text{pred}} \in \hat{\mathcal{P}}_{\text{pred}}$, and for some $\alpha \in [0, 1]$. It can be interpreted as the smallest collection of outputs y_* that have a high probability of being the correct ones for the new input x_* , according to all the distributions in $\hat{\mathcal{P}}_{\text{pred}}$. Or equivalently, we can say that the correct output for the new input x_* belongs to IHDR $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ with lower probability of at least $1 - \alpha$, a probabilistic guarantee for the set of outputs generated by CBDL. Notice also that the size of $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ is an increasing function of both (predictive) AU and EU. As a consequence, it is related, but it is not equal, to the (predictive) AU the agent faces. If we want to avoid to perform the procedure only to discover that $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ is “too large”, then we can add an “AU check” after **Step 5**. This, together with computing $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ in a classification setting, is explored in Caprio et al. [236, Appendices G and H].

As we have seen in Section 55.2.2, we find $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$ by taking the union of the $K \times S$ HDRs $R_\alpha(\hat{P}_{k,s}^{\text{pred}})$ of the extreme elements $\text{ex}\hat{\mathcal{P}}_{\text{pred}}$ of $\hat{\mathcal{P}}_{\text{pred}}$. The HDR of a well-known distribution (for instance, a Normal) can be routinely obtained in R, e.g. using package `HDInterval` [856].

We conclude this Section with two remarks. First, we point out how CBDL does not depend on the method used to approximate the posterior and the predictive distributions: Variational Inference can be substituted by other approaches. CBDL can also be easily adapted to other TU, AU, and EU measures, as long as the measure chosen for the total uncertainty is bounded. Second, **Step 6** can be effortlessly modified so that CBDL produces the collection of outputs having the highest lower density. This is in line with the Naive Credal Classifier theory [341, 1829]. In this case, we forego control on the accuracy level $1 - \alpha$ of the output region produced by CBDL. It is implemented as follows. In the modified version of **Step 6**, we compute

$$\arg \max_{y_* \in \mathcal{Y}} \min_{k,s} \hat{p}_{k,s}^{\text{pred}}(y_*), \quad (55.10)$$

where $\hat{p}_{k,s}^{\text{pred}}$ is the pdf of the VI-approximated predictive distribution $\hat{P}_{k,s}^{\text{pred}}$. If such $\arg \max$ is not a singleton, and the user is set on CBDL outputting a unique value y_* for the new input x_* , they can select one element uniformly at random from the $\arg \max$.

55.3.2 Theoretical Properties of CBDL

Working with credal sets makes CBDL more robust to distribution misspecification and shifts than single BNNs. To see this, we present the following general result, and then we apply it to our case.

¹⁵Curiously, a similar motivation was brought forward by Pseudo-Dionysius the Areopagite in favour of the use of sacred images in the Christian tradition [1172]. While they do not capture the essence of God, these defective approximations help the believer’s thought to elevate.

Let Π' be an FGCS as in Remark 55.2, and consider a probability measure Ψ such that $\Psi \notin \Pi'$.

Proposition 55.3. *Call d any metric and div any divergence on the space of probability measures of interest. Let $d(\Pi', \Psi) := \inf_{P' \in \Pi'} d(P', \Psi)$ and $\text{div}(\Pi' \| \Psi) := \inf_{P' \in \Pi'} \text{div}(P' \| \Psi)$. Then, for all $P' \in \Pi'$, $d(\Pi', \Psi) \leq d(P', \Psi)$ and $\text{div}(\Pi', \Psi) \leq \text{div}(P' \| \Psi)$.*

Proposition 55.3 holds if Π' is any set of probabilities, not just an FGCS.¹⁶ In Caprio et al. [236, Appendix I], the authors show that the above result still holds if the elements of Π' and Ψ are defined on Euclidean spaces having different dimensions [225, 233].

Let us now apply Proposition 55.3 to CBDL. Suppose that, when designing a single BNN, an agent chooses likelihood $\mathbf{L}$, while when implementing CBDL, they specify in **Step 2** a finite set of likelihoods $\text{ex}\mathcal{P}_{\text{lik}} = \{L_s^{\text{ex}}\}_{s=1}^S$, $S \geq 2$, and then let the induced credal set $\mathcal{P}_{\text{lik}} = \text{Conv}(\text{ex}\mathcal{P}_{\text{lik}})$ represent their uncertainty around the sampling model. Assume that $\mathbf{L} \in \text{ex}\mathcal{P}_{\text{lik}}$. This means that when designing the single BNN, the agent chooses arbitrarily which of the elements of $\text{ex}\mathcal{P}_{\text{lik}}$ to use. Suppose also that the “oracle” data generating process L° is different from $\mathbf{L}$, $L^\circ \neq \mathbf{L}$, so that we are actually in the presence of likelihood misspecification. Then, we have two cases. (1) If the true sampling model L° belongs to $\mathcal{P}_{\text{lik}}$, then the distance – measured via a metric or a divergence – between $\mathcal{P}_{\text{lik}}$ and L° is 0, while that between $\mathbf{L}$ and L° is positive. (2) If $L^\circ \notin \mathcal{P}_{\text{lik}}$, then the distance between $\mathcal{P}_{\text{lik}}$ and L° is no larger than the distance between $\mathbf{L}$ and L° , no matter (i) which metric or distance we use (Proposition 55.3), (ii) whether or not L° and the elements of $\mathcal{P}_{\text{lik}}$ are defined on the same Euclidean space [236, Lemma 17]. A visual representation is given in Figure 55.5. A similar argument holds if part of the training dataset $\mathcal{D}$ is generated by one distribution L_1° , and the remaining part is generated by L_2° , a situation that we can categorise as distribution shift. Even if – in the case of a single BNN – the user were able to elicit exactly $\mathbf{L} = L_1^\circ$, the likelihood would still be misspecified for the data that come from L_2° . This shortcoming is overcome by a credal set approach, in which either both oracle distributions L_1° and L_2° belong to the credal set $\mathcal{P}_{\text{lik}}$, or the distance between $\mathcal{P}_{\text{lik}}$ and either of them is no larger than that between the single chosen one $\mathbf{L}$, and L_1° and L_2° .

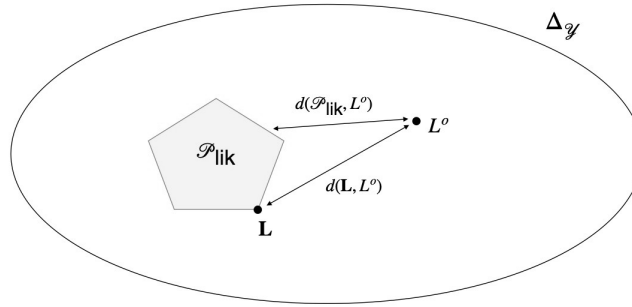


Figure 55.5: CBDL is more robust to distribution shifts than single BNNs. Here $\mathcal{P}_{\text{lik}}$ is the convex hull of five plausible likelihoods, and d denotes a generic metric on the space $\Delta_{\mathcal{Y}}$ of probabilities on $\mathcal{Y}$. We see how $d(\mathcal{P}_{\text{lik}}, L^\circ) < d(\mathbf{L}, L^\circ)$; if we replace metric d by a generic divergence div , the inequality would still hold.

Let us add a remark here. Assume that the “oracle” prior P° is in the prior credal set $\mathcal{P}_{\text{prior}}$ and that the “oracle” likelihood L° is in the likelihood credal set $\mathcal{P}_{\text{lik}}$. Then, it is immediate to see that the “oracle” posterior $P^\circ(\cdot | \mathcal{D})$ belongs to the posterior credal set $\mathcal{P}_{\text{post}}$. Naturally, this does not imply that the posterior credal set collapses to $P^\circ(\cdot | \mathcal{D})$. In general, it is unlikely that a finite amount of data is able to completely annihilate all the epistemic uncertainty faced by the agent. What may happen is that if the training set is large enough, $\mathcal{P}_{\text{post}}$ may be inscribed in a ball of small radius around $P^\circ(\cdot | \mathcal{D})$. This does not mean that we suffer from under-confidence due to larger-than-necessary epistemic uncertainty. Rather, the relative epistemic uncertainty (measured by the difference between prior and posterior uncertainty, divided by the prior uncertainty) drops significantly. In addition, working with sets of prior and likelihoods allows us to hedge against prior and likelihood misspecification, a consequence of Proposition 55.3.

55.4 Experiments

From the previous Sections, it is clear that CBDL improves on the uncertainty quantification capabilities of single BNNs.¹⁷ CBDL allows for a better quantification of predictive AU because of its robustness to misspecification

¹⁶More in general, Proposition 55.3 holds for any type of function f , not just metrics or divergences, because Proposition 55.3 essentially only applies the definition of the infimum.

¹⁷This includes empirical Bayes methodologies [927].

stemming from Proposition 55.3. In addition, as we pointed out before (in Section 55.2.1), predictive EU cannot be obtained in a theoretically principled manner from a single BNN [512, 804]. Here, we show that these theoretical arguments are also backed by empirical evidence.

55.4.1 (Predictive) Uncertainty Quantification

Distribution shifts can introduce uncertainties in a system, which in turn can render the predictions meaningless. This can be due to naturally occurring corruptions, as introduced in Hendrycks and Dietterich [733] for image datasets. The authors introduced 18 different noise types, which can be varied across 5 different severity levels, ranging from low (severity = 1), to medium (severity = 2, 3), and high (severity = 4, 5). The intuition is that in the current context, increasing the noise severity should generally result in higher uncertainty. We evaluate the CBDL method on 4 standard image datasets, CIFAR-10 [934], SVHN [1232], Fashion-MNIST [1785], and MNIST [993]. We use a slightly different set of perturbations than those introduced in Mu and Gilmer [1203] for gray-scale images like MNIST and Fashion-MNIST. Additionally, we perform cross domain testing for each dataset, where we expect the predictive uncertainties to be higher. We implement and train a Resnet-20 Bayesian Neural Network model inside the library Bayesian-torch [927]. For each dataset, we train 4 different networks initialised with different seeds on the prior and with the same architecture. This corresponds to eliciting a prior FGCS $\mathcal{P}_{\text{prior}}$ such that $\text{ex}\mathcal{P}_{\text{prior}} = \{P_1^{\text{ex}}, \dots, P_4^{\text{ex}}\}$, so that K in **Step 1** of Algorithm 39 is equal to 4, and a likelihood FGCS that is a singleton, $\mathcal{P}_{\text{lik}} = \text{ex}\mathcal{P}_{\text{lik}} = \{L\}$, so that S in **Step 2** of Algorithm 39 is equal to 1. We use a learning-rate of 0.001, batch-size of 128, and train the networks using Mean-Field Variational Inference for 200 epochs. The inference is carried out by performing multiple forward passes through parameters drawn from the posterior distribution. We used 20 Monte Carlo (MC) samples in the experiments (MC methods were introduced in Part I of this book).

Baselines. In Krishnan and Tickoo [926], for a single BNN output, the predictive distribution is obtained through multiple stochastic forward passes on the network while sampling from the weight posteriors using Monte Carlo estimators. In their work, the authors define the overall entropy and the entropy of the predictive distribution as the *predictive entropy* [926, Equation (C.2)]. This quantity captures a combination of predictive aleatoric and epistemic uncertainties. Additionally, they define the *mutual information* between weight posterior and predictive distribution as the predictive epistemic uncertainty. Finally, the predictive aleatoric uncertainty is defined as the *expected entropy*. The predictive epistemic and aleatoric uncertainties sum up to the predictive entropy. In the experiments, we use these definitions to compute the respective quantities.

In Cobb et al. [316], the authors consider different BNNs, but instead of keeping them separate and use them to build a predictive credal set, they average them out. Similar to theirs, we elicit the following procedure, that we call ensemble of BNNs (EBNN).

Consider $R \in \mathbb{N}_{\geq 2}$ different BNNs, and compute the posterior distribution on the parameters. They induce R predictive distributions on the output space $\mathcal{Y}$, each having mean μ_r and variance σ_r^2 , $r \in \{1, \dots, R\}$. We call *EBNN distribution* P_{ens} a Normal having mean $\mu_{\text{ens}} = 1/R \sum_{r=1}^R \mu_r$ and covariance matrix $\sigma_{\text{ens}}^2 I$, where $\sigma_{\text{ens}}^2 = 1/R \sum_{r=1}^R \sigma_r^2 + 1/(R-1) \sum_{r=1}^R (\mu_r - \mu_{\text{ens}})^2$.

Following Cobb et al. [316], for EBNN we posit that $1/R \sum_{r=1}^R \sigma_r^2$ captures the (predictive) aleatoric uncertainty associated with P_{ens} , and $1/(R-1) \sum_{r=1}^R (\mu_r - \mu_{\text{ens}})^2$ captures the (predictive) epistemic uncertainty associated with P_{ens} .¹⁸ For CBDL, we look at the value of $\text{AU}(\hat{\mathcal{P}}_{\text{pred}}) = \underline{H}(\hat{\mathcal{P}}_{\text{pred}})$ from (55.8), and at the lower bound $\overline{H}(\hat{\mathcal{P}}_{\text{pred}}) - \underline{H}(\hat{\mathcal{P}}_{\text{pred}})$ for EU($\hat{\mathcal{P}}_{\text{pred}}$) from (55.9). It is enough to focus on the lower bound because the upper bound is given by $\overline{H}(\hat{\mathcal{P}}_{\text{pred}}) - \underline{H}(\hat{\mathcal{P}}_{\text{pred}}) + \log(4)$, and $\log(4) \approx 0.6$ is a fixed value. Hence, the trend of both lower and upper bounds for EU($\hat{\mathcal{P}}_{\text{pred}}$) as the severity of corruption changes is the same.

Summary of the Results. Two scenarios are evaluated: an In-Distribution (ID) scenario with corruption by noises of increasing severity, and an Out-Of-Distribution (OOD) assessment. The following main points summarise our findings:

1. (ID Evaluation) For both CBDL and the method in Krishnan and Tickoo [926], for increasing severities of noise corruption, predictive epistemic and aleatoric uncertainties increase. In comparison, the baseline EBNN method demonstrates a counterintuitive result, where the predictive aleatoric uncertainty decreases as the severity of corruption increases. The table of values can be found in Caprio et al. [236, Appendix N].
2. (ID Evaluation) To assess the utility of CBDL in downstream tasks, we analyse the accuracy versus rejection rate for all methods. Table 55.3 summarises the results. Across MNIST-C and Fashion-MNIST-C,

¹⁸Notice that in the experiments we retain the assumption that the EBNN distribution P_{ens} on $\mathcal{Y}$ has mean μ_{ens} and covariance matrix $\sigma_{\text{ens}}^2 I$, but we do not require that it is a Normal. That is because in the four image datasets that we consider, the output space $\mathcal{Y}$ is finite.

CBDL exhibits a higher average accuracy across the range of rejection rates. For the remaining datasets, it demonstrates results comparable to the baseline (see Section 55.4.1 for evaluation details)

- (OOD Evaluation) To evaluate OOD detection, models trained on a particular dataset are tested on the other three datasets with the goal of analysing the behavior of the predictive uncertainties. Tables 55.4 and 55.5 summarise the results. Across all the evaluated datasets, CBDL had the most consistent behavior of exhibiting a large increase in predictive AU, relative to the increase in predictive EU when tested on an OOD dataset. In comparison, EBNN shows a decrease in predictive AU, and the single BNN shows inconsistent behavior depending on the dataset.

	CIFAR-10C			MNIST-C			Fashion MNIST-C			SVHN	
Severity	BNN	Ensemble of BNN	CBDL	BNN	Ensemble of BNN	CBDL	BNN	Ensemble of BNN	CBDL	BNN	Ensemble of BNN
1	0.754	0.983	0.982	0.645	0.919	0.929	0.525	0.730	0.748	0.698	0.913
2	0.761	0.947	0.946	0.680	0.879	0.896	0.537	0.655	0.641	0.703	0.843
3	0.749	0.906	0.907	0.617	0.804	0.841	0.442	0.498	0.529	0.669	0.764
4	0.733	0.838	0.836	0.639	0.713	0.737	0.332	0.421	0.489	0.608	0.586
5	0.661	0.734	0.734	0.404	0.495	0.539	0.300	0.351	0.391	0.455	0.378
Mean	0.732	0.882	0.881	0.597	0.762	0.788	0.433	0.531	0.560	0.627	0.697

Table 55.3: In-Distribution evaluation on noise corruptions of increasing severity. Average area under accuracy vs rejection rate curve. Higher numbers indicate overall better accuracy over a range of rejection rates.

		CBDL		Ensemble of BNN		BNN	
		Epistemic	Aleatoric	Epistemic	Aleatoric	Epistemic	Aleatoric
CIFAR	In Dist-Clean	0.102	0.031	0.008	0.076	0.151	0.179
	MNIST	0.184 (1.801)	0.142 (4.626)	0.021 (2.723)	0.044 (0.572)	0.445 (2.95)	0.772 (4.312)
	Fashion MNIST	0.183 (1.791)	0.147 (4.808)	0.022 (2.822)	0.042 (0.557)	0.431 (2.861)	0.811 (4.53)
	SVHN	0.183 (1.789)	0.141 (4.606)	0.019 (2.421)	0.046 (0.608)	0.394 (2.612)	0.647 (3.616)
MNIST	In Dist-Clean	0.188	0.057	0.012	0.063	0.198	0.071
	CIFAR	0.185 (0.986)	0.167 (2.94)	0.023 (1.948)	0.042 (0.67)	0.821 (4.155)	0.372 (5.124)
	Fashion MNIST	0.162 (0.866)	0.187 (3.287)	0.019 (1.618)	0.035 (0.564)	0.919 (4.654)	0.461 (6.496)
	SVHN	0.190 (1.015)	0.168 (2.963)	0.022 (1.847)	0.043 (0.69)	0.923 (4.674)	0.297 (4.183)
Fashion MNIST	In Dist-Clean	0.121	0.029	0.007	0.075	0.197	0.351
	CIFAR	0.205 (1.705)	0.104 (3.561)	0.022 (3.178)	0.050 (0.667)	0.408 (2.073)	0.221 (0.629)
	MNIST	0.165 (1.367)	0.182 (6.192)	0.021 (3.069)	0.033 (0.443)	0.909 (4.618)	0.328 (0.932)
	SVHN	0.218 (1.81)	0.109 (3.742)	0.026 (3.827)	0.046 (0.618)	0.598 (3.036)	0.305 (0.867)
SVHN	In Dist-Clean	0.061	0.01	0.004	0.083	0.097	0.107
	CIFAR	0.181 (2.992)	0.121 (11.778)	0.026 (6.029)	0.040 (0.479)	0.655 (6.765)	0.716(6.69)
	MNIST	0.198 (3.272)	0.063 (6.154)	0.024 (5.495)	0.056 (0.675)	0.329 (3.393)	0.336 (3.139)
	Fashion MNIST	0.199 (3.288)	0.113 (10.984)	0.026 (6.002)	0.041 (0.491)	0.103 (6.375)	0.125 (6.178)

Table 55.4: Out-Of-Distribution evaluation. Across all datasets, when tested on datasets outside of the training dataset, CBDL demonstrated the most consistent behavior in terms of predictive uncertainty quantification. The quantity in the parenthesis is the magnitude of increase relative to the quantity when tested on the clean dataset.

In-distribution Evaluation

Overall, CBDL demonstrates the most consistent behavior in terms of increasing levels of predictive epistemic and aleatoric uncertainties as the severity of noise corruption increases. The full table of results can be found in Caprio et al. [236, Appendix N]. In the experiments, for CIFAR-10 the single BNN also exhibits similarly consistent behavior. In MNIST and Fashion-MNIST, the BNN seems to be more consistent than CBDL, whereas for SVHN it seems to show a more inconsistent behavior. These results allow us to conclude that CBDL is comparable if not better than BNN in terms of In-Distribution behavior.

Accuracy vs Rejection Rate. As the quantities pertaining to different approaches are not comparable, we perform an additional evaluation comparing the accuracy vs the rejection rate of a given method. In this task, for a given uncertainty threshold, if the uncertainty quantity exceeds such a threshold, then the prediction is rejected. Of those not rejected, the accuracy is computed. A range of uncertainty thresholds is tested for each method and plotted. Figure 55.6 depicts an example of the plot. There, for lower severities (e.g. 1-2) of Gaussian blur

		CBDL		Ensemble of BNN		BNN	
		Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC
CIFAR	MNIST	0.905	0.826	0.805	0.104	0.860	0.929
	Fashion	0.920	0.836	0.816	0.093	0.808	0.937
	MNIST						
	SVHN	0.851	0.823	0.789	0.117	0.826	0.909
		CBDL		Ensemble of BNN		BNN	
		Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC
MNIST	CIFAR	0.930	0.912	0.827	0.040	0.835	0.882
	Fashion	0.956	0.935	0.804	0.030	0.929	0.960
	MNIST						
	SVHN	0.971	0.926	0.808	0.033	0.874	0.898
		CBDL		Ensemble of BNN		BNN	
		Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC
Fashion MNIST	CIFAR	0.933	0.724	0.862	0.121	0.703	0.770
	MNIST	0.925	0.881	0.770	0.085	0.962	0.933
	SVHN	0.972	0.830	0.865	0.066	0.855	0.904
		CBDL		Ensemble of BNN		BNN	
		Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC	Epi. AUROC	Alea. AUROC
SVHN	CIFAR	0.979	0.941	0.890	0.014	0.964	0.952
	MNIST	0.781	0.679	0.777	0.205	0.728	0.709
	Fashion	0.967	0.881	0.899	0.030	0.921	0.917
	MNIST						

Table 55.5: Out-Of-Distribution evaluation. AUROC for OOD detection is computed using both aleatoric and epistemic predictive uncertainty measures of the different approaches. Each model is trained on the training partition of the dataset and tested on the respective testing set of the OOD dataset. When using predictive AU, a single BNN performs relatively better than CBDL for OOD detection, though CBDL is still comparable and outperforms the single BNN when trained on Fashion-MNIST. When using predictive EU, CBDL outperforms the other approaches on most datasets.

noise, CBDL exhibits better performance and converges to 100% accuracy at a lower rejection rate. Intuitively, for the same range of rejection rates, a higher curve signifies better performance, as less samples are rejected while achieving a higher accuracy. However, for higher severities (above 3), the single BNN seems to show better performance. Even so, the overall rejection rate is high to achieve such accuracy. The entirety of the results can be found in Caprio et al. [236, Appendix P]. To quantify the differences, the areas under the curve are computed and averaged for a given noise severity level. Finally, this quantity is averaged across all severity levels. The results are shown in Table 55.3. As we can see, when EBNN outperforms CBDL in specific instances, the difference is minor, and when the opposite happens, the difference is significant. Over all datasets, CBDL outperforms a single BNN.

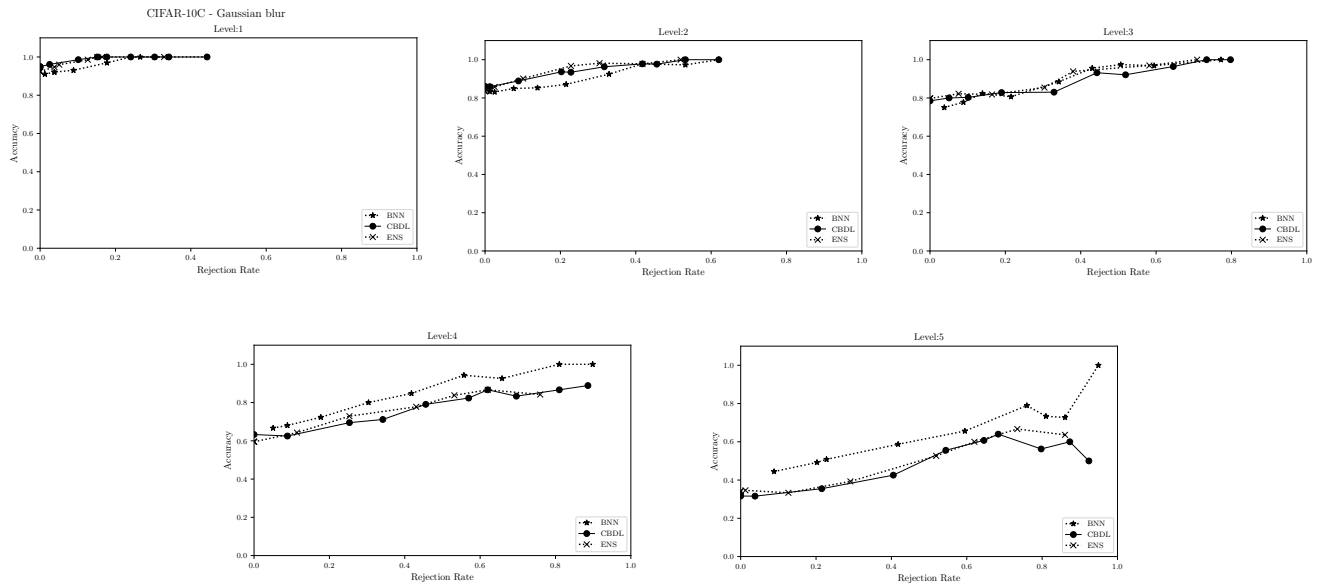


Figure 55.6: Accuracy vs Rejection Rate - CIFAR-10C Gaussian blur. For lower severities (e.g. 1-2) of Gaussian blur noise, CBDL exhibits better performance and converges to 100% accuracy at a lower rejection rate. For higher severities (above 3), the single BNN seems to show better performance. Even so, the overall rejection rate is high to achieve such accuracy. The ensemble method (EBNN) exhibits comparable performance in this specific instance.

Out-of-distribution Evaluation

In this final evaluation, we analyse the behavior of the three approaches (CBDL, EBNN, and BNN) when tested on an OOD dataset. For a given dataset, for example CIFAR, a model is trained on the dataset, and evaluated on the remaining datasets, i.e. MNIST, SVHN, and Fashion-MNIST. The relative increase in the average predictive EU and AU is computed compared to when an approach is tested on the In-Distribution testing set. The In-Distribution uncertainty quantities for each dataset are shown in Table 55.4. The relative magnitude of the increase in each quantity is shown in the parenthesis for each respective OOD dataset.

The baseline EBNN exhibits a counterintuitive behavior, viz. a decrease in predictive AU. Although this behavior is consistent across the different dataset combinations, this would be an undesirable quality for OOD detection scenarios.

The single BNN exhibits inconsistent behavior in terms of the relative increase or decrease in magnitude. For CIFAR-10, the predictive AU increases by a magnitude larger than the predictive EU, while for SVHN the increase is smaller. Moreover, for Fashion-MNIST the predictive AU decreases, and for MNIST the behavior is inconsistent.

Only CBDL demonstrates a stable, consistent increase in predictive AU, which is greater in magnitude relative to the increase in predictive EU. These results demonstrate a clear improvement in performance by CBDL.

We also give results pertaining AUROCs for the OOD detection performance, presented in Table 55.5. Each group of results shows which dataset the models are trained on, and which they are tested on. We report AUROC when using both the predictive EU and AU of each approach. In general, the baseline EBNN method performs the worst in all cases. When using predictive AU, a single BNN performs relatively better than CBDL for OOD detection, though CBDL is still comparable and outperforms it when trained on Fashion-MNIST. When using predictive EU, CBDL clearly outperforms the other approaches on most datasets. This is in line with other works where predictive EU is considered important for OOD detection [879]. We conjecture that this may be due to the fact that a single BNN is not able to gauge predictive EU properly. Hence, the single BNN flags an instance as OOD when it comes from the “tail” of the distribution; this is well-captured by (predictive) aleatoric uncertainty. On the contrary, CBDL is able to gauge predictive EU properly, and hence it is able to capture when OOD happens by looking at the “disagreement” between the elements of the predictive credal set, captured by the difference between upper and lower entropy.

55.5 Conclusion and Open Problems

We presented CBDL, a procedure that can be seen as a non-condensed, uncountably infinite ensemble of BNNs, carried out using only finitely many elements. It allows to distinguish between predictive AU and EU, and to quantify them. We showed how it can be used to specify a set of outputs – the IHDR – that enjoys probabilistic guarantees.

A region that improves on $IR_\alpha(\hat{\mathcal{P}}_{\text{pred}})$, meaning that it would be tighter, is $IR'_\alpha(\hat{\mathcal{P}}_{\text{pred}}) = \{y \in \mathcal{Y} : \hat{p}^{\text{pred}}(y) \geq \hat{p}^\alpha\}$, where $\hat{p}^{\text{pred}} := \min_{k,s} \hat{p}_{k,s}^{\text{pred}}$, and $\hat{p}^\alpha$ is the largest constant such that $\hat{P}^{\text{pred}}[y \in IR'_\alpha(\hat{\mathcal{P}}_{\text{pred}})] \geq 1 - \alpha$. The problem with IR'_α is that, while the highest density regions $R_\alpha(\hat{\mathcal{P}}_{k,s}^{\text{pred}})$ associated with the predictive distributions in $\text{ex}\hat{\mathcal{P}}_{\text{pred}}$ can be computed using off-the-shelf tools, calculating $\hat{p}^{\text{pred}}$ and $\hat{p}^\alpha$ would have been much more computationally expensive. In addition, it would have required to come up with a new technique to find $\hat{p}^{\text{pred}}$ and $\hat{p}^\alpha$. In turn, this remains an open problem. Another interesting endeavour is to derive a credal-evidential approach, that could allow the user to bypass the bottleneck in Algorithm 39 by merging imprecise probabilistic and evidential methods [29, 260, 413, 414, 1491].

In the future, CBDL should also be related to Bayesian Model Selection (BMS) [623], e.g. via a finite number of regularised horseshoe priors [623, Section 3.2] as extreme elements of the prior credal set, to induce shrinkage while hedging against prior misspecification. For more open problems, see Caprio et al. [236, Section 6].

Chapter 56

Bayesian Reinforcement Learning

Reinforcement learning asks a central question: how should an agent act while it is still learning about its environment so that rewards are maximised over time? The challenge appears across domains. A robot must learn locomotion without falling. A trading system must try new strategies without risking ruin. A clinical assistant must improve recommendations while caring for current patients. The tension between learning and performance is the theme of this chapter.

At the heart of this challenge lie two competing objectives. **Exploration** refers to selecting actions that gather information about unknown aspects of the environment. **Exploitation** refers to selecting actions that are expected to yield high rewards based on current knowledge. Balancing these two objectives is the central problem in learning under uncertainty. This exploration-exploitation trade-off is the core challenge that distinguishes learning problems from pure optimisation. Standard Reinforcement Learning (RL) typically defines performance with respect to the true environment, but during learning the agent does not know the transition dynamics or reward function that make this objective well defined. As a result, exploration is often added externally through heuristics such as random actions, count-based bonuses, or optimism. These mechanisms can be useful, but they do not directly represent what the agent does and does not know. Bayesian Reinforcement Learning (BRL) makes this uncertainty explicit by maintaining beliefs over plausible environments and using those beliefs to guide decisions. Uncertainty is therefore not only a source of error to be eliminated, but a resource for deciding which actions are worth taking. Consider a hospital delivery robot that moves among patient rooms, closets, and restricted areas. Heuristic exploration can be disruptive and uninformative. A Bayesian view directs exploration to places where learning will most improve future performance while avoiding actions that add little information. In short, experiments are chosen for what they will change downstream.

Not all uncertainty merits exploration. Some can be reduced by collecting data and therefore warrants directed exploration, other forms are inherent and cannot be reduced by collecting more data.

56.0.1 Why Traditional Exploration Often Fails

A common exploration strategy, ϵ -greedy, selects random actions with fixed probability. It treats all unknowns as equally promising and ignores where information matters. Optimism in the face of uncertainty adds bonuses for rarely visited states, which improves coverage but still does not measure the value of what is learned or the cost of risky probes.

Such methods can be costly in risky domains. A hospital robot driven by ϵ -greedy may enter operating theatres at the wrong time, disrupting procedures without learning anything useful. A medical recommender might randomise treatments in cases where learning yields little benefit. A Bayesian approach prioritizes experiments that can change future choices and avoids those that are unlikely to influence policy.

56.0.2 Historical Development: From Bandits to Full Environments

The Bayesian view of exploration under uncertainty traces to Thompson [1603], who balanced helping current patients with learning for future patients. This led to the multi-armed bandit model and Thompson sampling, which maintains beliefs over payoffs and samples actions according to those beliefs. Briefly, in the multi-armed bandit problem an agent repeatedly selects among K actions (arms), each yielding stochastic rewards from an unknown distribution, seeking to maximise cumulative reward while learning which arms are best.

RL extends bandits by introducing a state variable s . Actions a modify the current state, yield a reward r , which couples exploration with future action combinations. Duff [460] formalized this in the Bayes-adaptive MDP,

where beliefs about the unknown environment are part of the state. The environment represents the external system with which the agent interacts, generating state transitions and rewards in response to actions. The formulation is complete but computationally demanding, which motivates the approximations developed in this chapter. A useful bridge is posterior sampling for RL, which samples a Markov decision process (MDP) from the current belief, solves it, then executes the resulting policy for a period before resampling, thereby turning belief updates into an implicit exploration schedule. The policy defines the agent's behaviour by assigning a probability distribution over actions for each state.

56.0.3 The Bayesian Framework: Beliefs and Updates

BRL represents knowledge about the environment with probability distributions. Let the unknown environment be a random variable $\mathcal{M}$ with prior $p(\mathcal{M})$. As experience accrues we update beliefs via Bayes rule,

$$p(\mathcal{M} \mid \text{experience}) = \frac{p(\text{experience} \mid \mathcal{M}) p(\mathcal{M})}{p(\text{experience})}. \quad (56.1)$$

This turns experience into updated beliefs and makes uncertainty explicit and operable.

Experience can be written as a trajectory $\tau_t = \{s_0, a_0, r_0, s_1, \dots, s_t\}$ with likelihood

$$p(\tau_t \mid \mathcal{M}) = \rho_{\mathcal{M}}(s_0) \prod_{i=0}^{t-1} P_{\mathcal{M}}(s_{i+1} \mid s_i, a_i) p_{\mathcal{M}}(r_i \mid s_i, a_i), \quad (56.2)$$

which reflects the Markov property.

56.0.4 The Mathematical Structure of Sequential Decision Making

We model RL problems as MDPs with state space $\mathcal{S}$, action space $\mathcal{A}$, transitions $P(s' \mid s, a)$, rewards $r(s, a)$, discount $\gamma \in [0, 1)$, and initial distribution ρ_0 . The policy $\pi(a \mid s)$ maximises expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Value functions describe optimal behaviour. The action value function satisfies

$$Q^{\pi}(s, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi}(s') \quad \text{with} \quad V^{\pi}(s) = \sum_a \pi(a \mid s) Q^{\pi}(s, a). \quad (56.3)$$

When the MDP is known, dynamic programming solves these equations. In BRL expectations are taken under beliefs rather than a single model, which couples planning with learning and makes value a function of both physical state and knowledge state.

56.0.5 Illustrative Example: Hazardous Navigation

A hospital robot in a 10×10 grid begins with uncertain room types: patient rooms, supply closets, restricted areas, or open corridors. Each room type has different transition dynamics and reward structures that the robot must learn through experience. As it navigates and observes outcomes, it updates its beliefs about room categories using Bayesian inference. Observations that confirm or contradict expectations shift probability mass between hypotheses; after repeated visits, the posterior concentrates around the true room type, while unvisited locations retain high uncertainty.

The core challenge is deciding how to act while simultaneously learning the environment. At each decision point, the robot evaluates actions along two dimensions. First, it computes expected immediate return under current beliefs by averaging over possible environment configurations weighted by their posterior probability. Second, it estimates how much each action will reduce uncertainty about the environment, and how this knowledge reduction translates to better decisions in future episodes. From a doorway the robot can test an unknown room or take a safe corridor. The unknown room may have low immediate reward but high expected information value, which can shorten many future routes by revealing a fast passage. When time is critical it exploits reliable paths; when time allows it explores regions where resolving uncertainty is most useful. Because beliefs retain variance, BRL can reason about risk and reduce the chance of rare but costly outcomes through targeted information gathering.

Suppose the robot reaches a junction with two options: a corridor whose properties are well understood from many previous traversals, or an unexplored room whose type remains uncertain. Taking the known corridor provides predictable progress toward the goal. Entering the unknown room provides information about its type, which affects optimal routing decisions across all future episodes. If the room turns out to contain a shortcut, the robot can exploit this discovery repeatedly. If it is a restricted area, the robot learns to avoid it. The value

of exploration depends on how many future decisions will benefit from resolving this uncertainty. This trade-off between immediate performance and information gathering depends on temporal context. When the robot must deliver urgent medication, exploiting known reliable paths maximises the probability of timely delivery. During routine supply transport with relaxed time constraints, exploring uncertain regions provides information that improves long-term efficiency. The optimal balance between exploration and exploitation emerges from explicitly modelling how observations today affect decision quality tomorrow.

The maintained probability distributions also enable reasoning about risk. Rather than committing to a single best guess about room types, the robot tracks uncertainty through posterior variance. High variance in a region's posterior distribution indicates that outcomes in that region are unpredictable under current knowledge. The robot can identify actions where this uncertainty creates risk of rare but costly outcomes, such as accidentally entering an operating theatre during a procedure. By reasoning about the full distribution rather than only expected values, the robot distinguishes between low expected reward due to genuine low value versus low expected reward due to high uncertainty about potentially high value. The subsequent sections formalize these concepts, beginning with exact Bayesian methods for problems where posterior distributions can be computed analytically, then progressing to approximate methods that scale to complex environments.

56.0.6 Information as a Resource

Actions generate reward now and information for later. One measure of expected information value from action a in state s is the expected Kullback-Leibler divergence between posterior and prior beliefs,

$$\mathcal{I}_t(s, a) = \mathbb{E}_{s', r}[\text{KL}(p(\mathcal{M} \mid \tau_t \cup \{s, a, r, s'\}) \parallel p(\mathcal{M} \mid \tau_t))]. \quad (56.4)$$

This quantity predicts how much a decision can change future behaviour and is the target that practical approximations try to mimic efficiently. Random or count-based exploration does not estimate this quantity, which explains wasted samples in regions that do not influence policy.

56.1 Exact Bayesian methods: the tabular foundation

Exact Bayesian inference provides a theoretical gold standard for RL under uncertainty. In tabular domains with conjugate priors, we can maintain exact posterior distributions and compute optimal exploration strategies without approximation. These methods establish fundamental principles that all subsequent approximations attempt to preserve while scaling to more complex domains.

This section develops three foundational approaches. Posterior sampling (PSRL) demonstrates how to sample from belief distributions to guide exploration. Bayes-Adaptive Monte Carlo Planner (BAMCP) shows how to plan optimally when exact dynamic programming becomes intractable. Bayesian Exploration Bonuses (BEB) illustrate how to approximate information value through explicit reward augmentation. Together, these methods establish the theoretical framework that guides all modern Bayesian RL algorithms.

56.1.1 Posterior Sampling for Reinforcement Learning (PSRL)

Posterior sampling provides the most direct implementation of Bayesian exploration principles. Rather than attempting to solve the computationally intractable Bayes-adaptive MDP exactly, PSRL samples a single environment from the posterior distribution and acts optimally within that sampled environment. This simple procedure captures the essence of uncertainty-driven exploration while remaining computationally tractable.

Mathematical Foundation

Consider a tabular MDP where states $\mathcal{S} = \{1, 2, \dots, S\}$ and actions $\mathcal{A} = \{1, 2, \dots, A\}$ are discrete and finite. The unknown transition probabilities $P(s' \mid s, a)$ and reward function $r(s, a)$ must be learned from experience.

We place conjugate priors over unknown parameters to enable exact Bayesian inference. For transition probabilities, we use Dirichlet priors:

$$P(\cdot \mid s, a) \sim \text{Dirichlet}(\alpha_{s,a,1}, \alpha_{s,a,2}, \dots, \alpha_{s,a,S}) \quad (56.5)$$

The hyperparameters $\alpha_{s,a,s'}$ represent prior counts for transitions from state s to state s' via action a . For rewards, we assume Gaussian distributions with known variance but unknown means:

$$r(s, a) \sim \mathcal{N}(\mu_{s,a}, \sigma_r^2) \quad (56.6)$$

with normal priors over the means:

$$\mu_{s,a} \sim \mathcal{N}(\mu_0, \sigma_0^2) \quad (56.7)$$

After observing trajectory data $\tau_t = \{s_0, a_0, r_0, s_1, \dots, s_t\}$, we update beliefs using conjugacy properties. Let $N_{s,a,s'}(t)$ denote the count of transitions from state s to state s' via action a up to time t . The posterior over transition probabilities becomes:

$$P(\cdot|s, a)|\tau_t \sim \text{Dirichlet}(\alpha_{s,a,1} + N_{s,a,1}(t), \dots, \alpha_{s,a,S} + N_{s,a,S}(t)) \quad (56.8)$$

For reward parameters, let $\bar{r}_{s,a}(t)$ denote the sample mean reward and $n_{s,a}(t)$ the visit count for state-action pair (s, a) . The posterior over reward means becomes:

$$\mu_{s,a}|\tau_t \sim \mathcal{N}\left(\frac{\sigma_r^2 \mu_0 + n_{s,a}(t) \sigma_0^2 \bar{r}_{s,a}(t)}{\sigma_r^2 + n_{s,a}(t) \sigma_0^2}, \frac{\sigma_r^2 \sigma_0^2}{\sigma_r^2 + n_{s,a}(t) \sigma_0^2}\right) \quad (56.9)$$

These posterior distributions capture our uncertainty about environment parameters. Initially, when visit counts are low, posteriors remain close to priors. As data accumulates, posteriors concentrate around observed frequencies and sample means.

Theoretical Guarantees

Osband et al. [1268] prove that PSRL achieves Bayesian regret $\tilde{O}(HS\sqrt{AT})$ where H is episode length, S is number of states, A is number of actions, and T is total time steps. This bound is order optimal, meaning no algorithm can achieve fundamentally better scaling without additional assumptions.

The $\sqrt{T}$ dependence indicates sublinear regret growth, so average per-step regret approaches zero as $T \rightarrow \infty$. The $S\sqrt{A}$ dependence reflects the exploration cost: larger state-action spaces require more exploration to identify optimal actions. The episodic commitment structure proves essential for these guarantees [1267].

Bayesian regret measures the expected cumulative performance gap between the algorithm and the optimal policy, with the expectation taken over the prior distribution over environments. In the episodic setting, let H denote the episode length, let E denote the number of episodes, and let $T = EH$ be the total number of interaction steps. If $s_{e,0}$ is the initial state of episode e , the Bayesian regret can be written as:

$$\text{BayesRegret}(T) = \mathbb{E}_{M \sim p(M)} \left[\sum_{e=1}^E \left(V_H^{*,M}(s_{e,0}) - \mathbb{E}_{\pi_e} \left[\sum_{h=0}^{H-1} r_{e,h} \mid M \right] \right) \right], \quad (56.10)$$

where M represents the true environment sampled from prior $p(M)$, $V_H^{*,M}(s_{e,0})$ is the optimal value function in environment M starting from initial state $s_{e,0}$, and the inner expectation is over trajectories generated by the algorithm's policy π .

The PSRL Algorithm

Algorithm 40 presents the complete PSRL procedure, implementing the theoretical framework developed by Strens [1569] and analyzed by Osband et al. [1268]. The algorithm operates episodically, committing to a single sampled environment for an entire episode before resampling—a design choice that proves necessary for achieving optimal regret guarantees [1267]. The episodic structure addresses a fundamental challenge: how to maintain temporal consistency in exploration strategies.

The algorithm decomposes into four distinct phases that repeat each episode. Phase 1 implements Bayesian learning by updating posterior distributions using conjugacy relationships. Phase 2 implements Thompson's principle: sample one plausible environment and act optimally within it. Phase 3 solves the standard MDP using value iteration. Value iteration is a dynamic programming algorithm that computes the optimal value function and policy for a known MDP by repeatedly applying the Bellman optimality update $V_{k+1}(s) = \max_a [r(s, a) + \gamma \sum_{s'} P(s'|s, a) V_k(s')]$ until convergence. In the context of PSRL, once an environment is sampled from the posterior distribution, value iteration solves this sampled MDP to obtain the optimal policy for that specific environment hypothesis. Phase 4 executes the policy and collects experience through interaction with the real environment. PSRL's exploration behaviour emerges from the interaction between posterior sampling and optimal decision-making according to the sampled environment hypothesis. When visit counts are low, posterior distributions remain diffuse with high variance. Sampling from these distributions produces diverse environment hypotheses, leading to exploratory policies. As evidence accumulates, posteriors concentrate around true parameter values, and sampled environments become increasingly similar to the true environment, leading to exploitation of accumulated knowledge.

Algorithm 40: Posterior Sampling for Reinforcement Learning (PSRL)

Data: Prior parameters $\alpha, \mu_0, \sigma_0^2$; episode length H ; number of episodes E

Initialize:

Set $N_{s,a,s'} = 0$ for all (s, a, s') // transition counts

Set $R_{s,a} = 0, n_{s,a} = 0$ for all (s, a) // reward sums, visit counts

for episode $e = 1, 2, \dots, E$ **do**

STEP 1: Update posterior beliefs from data

for each state-action pair (s, a) **do**

$\alpha_{s,a,s'}^{\text{post}} = \alpha_{s,a,s'} + N_{s,a,s'}$ for all s' // posterior Dirichlet parameters

if $n_{s,a} > 0$ **then**

$\tilde{r}_{s,a} = R_{s,a}/n_{s,a}$ // sample mean reward

 Update $\mu_{s,a}^{\text{post}}, \sigma_{s,a}^{\text{post}}$ via Eq. (56.9) // posterior mean and variance

else

$\mu_{s,a}^{\text{post}} = \mu_0, \sigma_{s,a}^{\text{post}} = \sigma_0$ // unvisited state-action pairs use priors

STEP 2: Sample one plausible environment

for each state-action pair (s, a) **do**

$\tilde{P}(\cdot|s, a) \sim \text{Dirichlet}(\alpha_{s,a,1}^{\text{post}}, \dots, \alpha_{s,a,S}^{\text{post}})$ // sample transitions

$\tilde{r}(s, a) \sim \mathcal{N}(\mu_{s,a}^{\text{post}}, (\sigma_{s,a}^{\text{post}})^2)$ // sample rewards

STEP 3: Solve MDP with sampled parameters

$\pi^{(e)} = \text{ValueIteration}(\tilde{P}, \tilde{r}, \gamma)$ // compute optimal policy

STEP 4: Execute policy and collect experience

$s = s_0$ // start at initial state

for step $h = 1$ to H **do**

$a = \pi^{(e)}(s)$ // select action from policy

 Execute action a , observe s' and r // interact with real environment

$N_{s,a,s'} = N_{s,a,s'} + 1$ // update transition count

$R_{s,a} = R_{s,a} + r$ // update reward sum

$n_{s,a} = n_{s,a} + 1$ // update visit count

$s = s'$ // transition to next state

56.1.2 Bayes-Adaptive Monte Carlo Planner

While PSRL provides optimal exploration guarantees, it requires exact value iteration that becomes intractable in large state spaces. Bayes-Adaptive Monte Carlo Planner [682] extends the posterior sampling principle to domains where exact planning is infeasible by combining root sampling with Monte Carlo Tree Search (MCTS).

Motivation and Computational Challenge

The exact Bayes-adaptive approach requires planning in the augmented state space (s, b) where b represents the current belief state over environments. This augmented space grows exponentially with the original state space size, making exact planning intractable even for moderate tabular domains.

BAMCP resolves this intractability through root sampling: instead of planning over the continuous belief space, the algorithm samples multiple environments from the current posterior distribution and performs standard MCTS planning within each sampled environment. This approach leverages the law of total expectation:

$$Q^*(s, a, b) = \mathbb{E}_{\mathcal{M} \sim b}[Q_{\mathcal{M}}^*(s, a)] \approx \frac{1}{K} \sum_{k=1}^K Q_{\mathcal{M}^{(k)}}^*(s, a), \quad (56.11)$$

where $Q^*(s, a, b)$ is the Bayes-optimal action-value function that represents the expected optimal Q-value under the posterior belief b over environments, $Q_{\mathcal{M}^{(k)}}^*(s, a)$ is the optimal Q-value for taking action a in state s under the specific sampled environment $\mathcal{M}^{(k)}$, $\mathcal{M}^{(k)}$ are independent samples from posterior belief b , and K is the number of sampled environments.

Convergence Properties

BAMCP inherits convergence guarantees from both MCTS and posterior sampling [683]. As simulation budget K increases, action value estimates converge to their true expectations under the posterior belief.

Theorem 56.1 (BAMCP Convergence). *Let $Q_b^*(s, a) = \mathbb{E}_{\mathcal{M} \sim b}[Q_{\mathcal{M}}^*(s, a)]$ denote the true expected action value under posterior belief b . Then with probability $1 - \delta$:*

$$|Q_K(a) - Q_b^*(s, a)| \leq \sqrt{\frac{V_{\max} \log(1/\delta)}{2n_a(K)}}, \quad (56.12)$$

where $Q_K(a)$ is BAMCP's estimate after K simulations, $n_a(K)$ is the number of times action a was selected, and $V_{\max} = R_{\max}/(1 - \gamma)$ is an upper bound on the discounted return when rewards are bounded by $R_{\max}$.

The convergence rate depends on how frequently each action is selected during simulations. Actions that appear optimal in many sampled environments receive more simulations through adaptive allocation of the Upper Confidence Bound (UCB) algorithm [61], leading to more accurate value estimates. The UCB algorithm balances exploration and exploitation by selecting actions that maximise $\bar{x}_a + \sqrt{\frac{2 \ln n}{n_a}}$, where $\bar{x}_a$ is the empirical mean reward, n is the total number of trials, and n_a is the number of times action a has been selected. This exploration bonus ensures that under-explored actions receive attention while favoring actions with high observed rewards.

Relationship to PSRL

BAMCP shares the root sampling principle with PSRL but differs in planning methodology and temporal commitment. PSRL commits to one sampled environment per episode and computes globally optimal policies using exact methods. BAMCP samples multiple environments per decision and uses MCTS to approximate optimal actions through simulation-based planning.

The choice between PSRL and BAMCP depends on problem characteristics. Use PSRL when exact planning is feasible, episodic structure is natural, and temporal consistency is important. Use BAMCP when state spaces are large, online planning is required, or computational budgets are limited. Both methods implement the same underlying Bayesian principle of sampling plausible environments and acting optimally within those environments.

The BAMCP Algorithm

Algorithm 41 presents the complete BAMCP procedure. The algorithm performs online planning at each decision point by sampling multiple environments and running MCTS simulations within each sampled environment. The algorithm decomposes into three phases that repeat for each simulation. Phase 1 samples one complete environment from the posterior belief using conjugate sampling procedures. Phase 2 executes one complete MCTS simulation in the sampled environment, treating it as the true environment. Phase 3 updates action value estimates and visit

Algorithm 41: Bayes-Adaptive Monte Carlo Planner

Data: Current state s , posterior belief b , simulation budget K

Data: Exploration constant c , rollout policy π_{rollout} , discount factor γ , horizon H

Initialize:

Set $Q(a) = 0$ and $N(a) = 0$ for all $a \in \mathcal{A}$

// action values

for simulation $k = 1, 2, \dots, K$ **do**

STEP 1: Sample environment from posterior

$\tilde{\mathcal{M}}^{(k)} \leftarrow \text{SampleEnvironment}(b)$

// e.g., PSRL-style c

STEP 2: MCTS simulation in sampled environment

$(a^{(k)}, R^{(k)}) \leftarrow \text{MCTS_SIM}(s, \tilde{\mathcal{M}}^{(k)}, c, \pi_{\text{rollout}}, \gamma, H)$

STEP 3: Update action statistics

$N(a^{(k)}) \leftarrow N(a^{(k)}) + 1$

$Q(a^{(k)}) \leftarrow Q(a^{(k)}) + \frac{R^{(k)} - Q(a^{(k)})}{N(a^{(k)})}$

// increment

Return $\arg \max_a Q(a)$

// action with high

counts using incremental averaging.

56.1.3 Bayesian Exploration Bonuses (BEB)

Although PSRL and BAMCP provide principled approaches to Bayesian exploration, they require episodic commitment or online planning that may be computationally expensive. Bayesian Exploration Bonuses [913] offer an alternative by augmenting immediate rewards with terms that approximate information value, enabling integration with standard value-based RL algorithms.

Information-Theoretic Foundation

BEB approximates information value using a computationally tractable heuristic. The theoretical foundation builds on expected information gain. Let $\mathcal{M}$ represent a specific environment instance and $\tau_t = \{s_0, a_0, r_0, s_1, \dots, s_t\}$ denote observed trajectory. Taking action a in state s provides information gain measured by expected change in posterior beliefs:

$$\mathcal{I}_t(s, a) = \mathbb{E}_{s', r \sim p(\cdot | s, a, \tau_t)} [\text{KL}(p(\mathcal{M} | \tau_t \cup \{s, a, r, s'\}) \| p(\mathcal{M} | \tau_t))] \quad (56.13)$$

For tabular MDPs with conjugate priors, information gain can be computed analytically. However, this computation remains expensive. BEB approximates information gain using a simple heuristic based on visit counts.

The Exploration Bonus Heuristic

BEB approximates expected information gain using the exploration bonus:

$$b_t(s, a) = \frac{\beta}{1 + n_{s,a}(t)}, \quad (56.14)$$

where $n_{s,a}(t)$ denotes the accumulated count for state-action pair (s, a) , including prior pseudo-counts when using the Dirichlet-belief notation of [913], and $\beta > 0$ controls the strength of exploration.

The $1/(1+n)$ decay reflects the BEB analysis: under Dirichlet beliefs, the effect of one additional observation on the posterior transition model decreases on the order of $1/n$. This makes BEB greedier than PAC-MDP bonuses such as MBIE-EB, which use $1/\sqrt{n}$ -type bonuses and therefore encourage more persistent exploration.

This implements the principle of “optimism in the face of uncertainty” [61]. State-action pairs with low visit counts receive large bonuses, making them appear artificially attractive and encouraging exploration. As visit counts increase, bonuses decrease, naturally shifting behaviour toward exploitation.

Integration with Temporal Difference (TD) Learning

BEB integrates with standard temporal difference learning algorithms by treating exploration bonuses as additional reward signals. The agent selects actions to maximise augmented Q-values:

$$a_t = \arg \max_a [Q_t(s_t, a) + b_t(s_t, a)], \quad (56.15)$$

where $Q_t(s, a)$ represents current value estimates learned via Q-learning, SARSA, or other methods.

The exploration bonus affects action selection but not the Q-learning update itself. This separation ensures Q-values learn about actual environmental rewards while bonuses guide exploration toward uncertain regions.

Theoretical Guarantees

BEB provides theoretical guarantees by showing that, after a polynomial number of exploratory steps, its policy is near-optimal relative to the optimal Bayesian policy for the current belief state [913].

Theorem 56.2 (BEB Regret Bound). *Let A_t denote the policy followed by BEB at time t , with $\beta = 2H^2$. Suppose the belief for a state-action pair is no longer updated once its total count exceeds $4H^3/\epsilon$. Then, with probability at least $1 - \delta$,*

$$V_H^{A_t}(b_t, s_t) \geq V_H^*(b_t, s_t) - \epsilon \quad (56.16)$$

for all but

$$O\left(\frac{|\mathcal{S}||\mathcal{A}|H^6}{\epsilon^2} \log \frac{|\mathcal{S}||\mathcal{A}|}{\delta}\right) \quad (56.17)$$

time steps.

This guarantee is not a standard regret bound against the optimal policy of a fixed true MDP. Instead, it states that BEB is near-optimal relative to the optimal Bayesian policy for the current belief state, except for a polynomial number of time steps. This distinction matters: BEB can use a greedier bonus than PAC-MDP algorithms because it aims to approximate Bayesian exploration rather than guarantee near-optimality for every possible underlying MDP.

56.1.4 Synthesis and Practical Guidance

These three methods represent different points in the trade-off between statistical rigor and computational efficiency.

PSRL provides the most principled approach when episodes have natural boundaries, planning in sampled environments is tractable, strong theoretical guarantees are important, and domains have moderate state-action spaces ($< 10^4$ pairs).

BAMCP works best when exact planning is intractable but simulation is feasible, online planning budgets are available at decision time, domains require sophisticated forward search, and belief state representations are complex but sampling is straightforward.

BEB offers practical advantages when real-time decision making is required, integration with existing value-based methods is desired, computational resources are severely constrained, and domain structure makes visit count approximations reasonable.

The key insight from exact methods is that optimal exploration requires commitment to hypotheses for extended periods [1267]. Random or myopic exploration fails because it cannot sustain consistent action sequences needed to test complex hypotheses about environment structure. Whether implemented through posterior sampling (PSRL), planning within sampled environments (BAMCP), or information-driven bonuses (BEB), this commitment principle guides all subsequent approximation methods.

56.2 Gaussian Process Methods: Nonparametric Bayesian RL for Continuous Spaces

Gaussian process (GP) methods extend BRL to continuous state and action spaces through nonparametric function approximation. Unlike linear methods that assume fixed basis functions, Gaussian processes define probability distributions over function spaces, enabling flexible representations that adapt to data complexity while maintaining principled uncertainty quantification.

This section examines GP-SARSA for value function learning, PILCO for model-based policy search, and briefly discusses GP-based Model Predictive Control (MPC) for safe planning under uncertainty. These methods establish the theoretical framework for nonparametric function approximation while providing practical algorithms that achieve data efficiency through uncertainty-guided exploration.

56.2.1 From parametric to Nonparametric Function Approximation

Moving beyond tabular methods, Gaussian process approaches increase flexibility by modelling value functions or dynamics directly in function space, at the cost of greater computational complexity.

Limitations of Parametric Approaches

Linear methods assume value functions can be written as $V(s) = \phi(s)^\top \theta$ where $\phi(s) \in \mathbb{R}^d$ represents feature vectors. This reduces parameters from $|\mathcal{S}|$ to d , enabling tractable learning when $d \ll |\mathcal{S}|$. However, if the chosen features cannot represent the true value function, no amount of data can overcome this representational bias.

Gaussian processes address this by placing priors over function spaces rather than parameter spaces, adapting their representation based on observed data. This flexibility comes at a computational cost: exact inference scales as $O(n^3)$ where n is the number of observations, compared to $O(d^2)$ for linear methods.

Gaussian Processes as Distributions over Functions

As introduced in Chapter 23, a Gaussian process is a Bayesian model that represents uncertainty over functions. It defines a distribution where any set of function values is jointly Gaussian, allowing us to predict both expected outputs and their uncertainty for new inputs.

56.2.2 GP-SARSA: Value Function Learning with Gaussian Processes

GP-SARSA extends temporal difference learning to continuous spaces using Gaussian processes to represent action-value functions. The method learns $Q(s, a)$ as a Gaussian process over the joint state-action space.

Given a transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$, the TD target is computed as:

$$y_t = r_t + \gamma \tilde{Q}(s_{t+1}, a_{t+1}), \quad (56.18)$$

where $\tilde{Q}(s_{t+1}, a_{t+1})$ is sampled from the GP posterior $\mathcal{N}(\mu_Q(s_{t+1}, a_{t+1}), \sigma_Q^2(s_{t+1}, a_{t+1}))$ for Thompson sampling exploration, or alternatively set to the posterior mean $\mu_Q(s_{t+1}, a_{t+1})$ for deterministic bootstrapping. The stochastic sampling variant enables natural exploration through posterior uncertainty, while the mean variant provides lower-variance targets at the cost of reduced exploration. This target forms a training pair (x_t, y_t) where $x_t = (s_t, a_t)$ represents the concatenated state-action input.

The GP posterior after observing n transitions provides:

$$\mu_Q(s, a) = \mathbf{k}^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \quad (56.19)$$

$$\sigma_Q^2(s, a) = k((s, a), (s, a)) - \mathbf{k}^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}, \quad (56.20)$$

where $\mathbf{k}$ contains covariances between (s, a) and observed state-action pairs, K represents the gram matrix of observed pairs, and $\mathbf{y}$ contains TD targets.

The uncertainty $\sigma_Q^2(s, a)$ enables exploration through Thompson sampling or upper confidence bounds. Actions are selected by:

$$a_t = \arg \max_a [\mu_Q(s_t, a) + \beta \sigma_Q(s_t, a)], \quad (56.21)$$

where $\beta > 0$ controls exploration intensity, balancing exploitation of high-value actions against exploration of uncertain regions.

Exploration mechanisms. GP-SARSA supports multiple exploration strategies leveraging posterior uncertainty. Thompson sampling draws $\tilde{Q}(s, a)$ from the GP posterior and acts greedily with respect to this sample, naturally trading off exploration and exploitation through posterior sampling. The UCB exploration, shown in the equation above, adds $\beta \sigma_Q(s, a)$ to the mean, with β controlling optimism about uncertain regions. The UCB strategy can be interpreted as optimism in the face of uncertainty: actions with high uncertainty receive exploration bonuses proportional to their standard deviation. These strategies share a common principle: uncertainty guides exploration toward regions where the agent can most improve its knowledge, enabling more efficient learning than ϵ -greedy approaches that explore uniformly regardless of epistemic uncertainty.

Algorithm 42: GP-SARSA with UCB Exploration

Data: Initial state s_0 , discount factor γ , exploration parameter β , observation noise σ_n^2

Data: Covariance function $k(\cdot, \cdot)$

Initialize dataset $\mathcal{D} = \emptyset$

Select initial action a_0 using exploration strategy

for $episode = 1, 2, \dots$ **do**

 Initialize state s_t , select action a_t

while *not terminal* **do**

 Execute action a_t , observe r_t and s_{t+1}

 Select $a_{t+1} = \arg \max_a [\mu_Q(s_{t+1}, a) + \beta \sigma_Q(s_{t+1}, a)]$

 Compute TD target $y_t = r_t + \gamma \mu_Q(s_{t+1}, a_{t+1})$

 Add transition to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t), y_t\}$

 Update GP posterior:

 Compute K with entries $K_{ij} = k((s_i, a_i), (s_j, a_j))$ for all pairs in $\mathcal{D}$

 For any (s, a) , compute $\mu_Q(s, a) = \mathbf{k}^\top (K + \sigma_n^2 I)^{-1} \mathbf{y}$

 For any (s, a) , compute $\sigma_Q^2(s, a) = k((s, a), (s, a)) - \mathbf{k}^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}$

$s_t \leftarrow s_{t+1}$, $a_t \leftarrow a_{t+1}$

56.2.3 PILCO: Probabilistic Inference for Learning Control

PILCO achieves sample-efficient learning through model-based policy search with Gaussian process dynamics models. The method learns probabilistic models of system dynamics and analytically propagates uncertainty

through time to optimize policies.

Probabilistic Dynamics Models

PILCO models state transitions as:

$$s_{t+1} = f(s_t, a_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_\epsilon), \quad (56.22)$$

where $f(\cdot, \cdot) \sim \mathcal{GP}(0, k_f)$ follows a Gaussian process and Σ_ϵ represents process noise covariance. When the state $s_t \in \mathbb{R}^D$ is a vector with D dimensions, PILCO learns D independent Gaussian process models $\{f_i\}_{i=1}^D$, one for each dimension of the next state: $s_{t+1,i} = f_i(s_t, a_t) + \epsilon_{t,i}$ where $f_i \sim \mathcal{GP}(0, k_{f_i})$ and $\epsilon_{t,i} \sim \mathcal{N}(0, \sigma_{\epsilon,i}^2)$.

Policy optimisation through Moment Matching

The policy optimisation objective maximises expected cumulative reward:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} r(s_t, a_t) \right], \quad (56.23)$$

where the expectation is taken over trajectories generated by policy π interacting with the learned dynamics model.

Policy parameterization and gradient computation. PILCO typically uses Gaussian policies $\pi(a|s) = \mathcal{N}(a; \mu_\pi(s), \Sigma_\pi)$ where the mean $\mu_\pi(s)$ is a parameterized function of state, often implemented as a radial basis function network or linear controller. Given a Gaussian belief over the current state $p(s_t) = \mathcal{N}(s_t; \mu_t, \Sigma_t)$, the action distribution is computed by marginalizing over state uncertainty. The resulting state distribution at $t+1$ after applying the GP dynamics is approximated through moment matching: the exact (non-Gaussian) distribution resulting from pushing the Gaussian state-action distribution through the nonlinear GP mean function is projected onto a Gaussian with matching first and second moments. Policy gradients $\nabla_\theta J(\pi_\theta)$ are computed by backpropagating through this chain of moment computations, enabling gradient-based optimisation without costly Monte Carlo rollouts. This analytical gradient computation is PILCO's key advantage, though it requires careful implementation of moment matching derivatives and is limited to scenarios where the Gaussian approximation remains accurate.

PILCO's key contribution is analytical computation of this expectation through moment matching. When the policy is Gaussian and the GP dynamics model uses specific kernel choices (such as squared exponential), the distribution over states at each time step can be approximated as Gaussian. By propagating only the mean and covariance of these state distributions through the GP dynamics model and reward function, PILCO computes expected returns and their gradients in closed form, avoiding expensive trajectory sampling. This analytical approach enables gradient-based policy optimisation with dramatically fewer environment interactions than model-free methods, because PILCO uses the learned GP model to simulate trajectories and compute policy gradients entirely in silico, only requiring actual environment interactions for collecting data to train the dynamics model.

56.2.4 Connection to Uncertainty Types and Exploration

As mentioned previously, in Bayesian methods, total uncertainty decomposes into two fundamental types: epistemic uncertainty and aleatoric uncertainty, the latter being irreducible uncertainty inherent to the environment's stochastic nature. This decomposition is key for principled exploration because only epistemic uncertainty justifies collecting additional data—aleatoric uncertainty persists regardless of how much data is collected and instead requires robust planning methods. Gaussian process methods preserve this fundamental uncertainty decomposition while extending to continuous spaces. The GP posterior variance $\sigma^2(x)$ captures epistemic uncertainty about function values that decreases as relevant data accumulates. This uncertainty exhibits distance-dependent decay near observations, growth in unexplored regions, and data-dependent reduction as informative observations accumulate.

The observation noise variance σ_n^2 represents aleatoric uncertainty reflecting environmental stochasticity and measurement errors that persist regardless of model accuracy. Distinguishing epistemic and aleatoric uncertainty enables appropriate exploration: epistemic uncertainty justifies additional data collection, while aleatoric uncertainty requires robust planning methods.

56.3 Neural Network Approximations: Scaling Bayesian RL to High-Dimensional Problems

The Gaussian process methods in the previous section provide principled uncertainty quantification but face computational challenges in high-dimensional spaces. Computing kernel matrices scales cubically with dataset size, and the inverse operations required for prediction become prohibitive for problems with thousands of observations. Neural network approximations address these limitations by replacing exact inference with scalable optimisation-based methods. These approaches maintain the core Bayesian principle—quantifying uncertainty to guide exploration—while sacrificing theoretical guarantees for practical applicability to complex domains like robotic control and game playing.

This section examines two complementary approaches to neural function approximation in Bayesian RL. Variational inference methods provide principled posterior approximations through optimisation, explicitly representing distributions over network parameters and computing gradients through the evidence lower bound. Ensemble methods approximate posterior uncertainty through multiple independently trained networks, sacrificing theoretical rigor for computational simplicity and robust empirical performance. Both approaches preserve the fundamental exploration mechanisms from earlier sections—Thompson sampling and upper confidence bounds—while enabling application to problems beyond the reach of exact methods.

56.3.1 From Exact Inference to Neural Approximation

To understand how neural networks fit into the Bayesian framework, consider the value function approximation problem. In earlier sections we represented $Q(s, a)$ exactly through tables or nonparametrically through Gaussian processes. Neural networks instead parameterize the value function as $Q_\theta(s, a)$ where $\theta \in \mathbb{R}^p$ denotes the weights and biases across all network layers. For a network with p parameters, this vector might contain millions of elements organized into weight matrices and bias vectors connecting successive layers of neurons. The Bayesian perspective treats these parameters as random variables with a prior distribution $p(\theta)$ rather than point estimates to be optimized. Given experience $\mathcal{D} = \{(s_i, a_i, y_i)\}_{i=1}^n$ where y_i represents target values (such as TD targets in temporal difference learning), we seek the posterior distribution:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \quad (56.24)$$

Here $p(\mathcal{D}|\theta)$ is the likelihood of observing the target values given parameters θ , $p(\theta)$ is the prior encoding our initial beliefs about reasonable parameter values, and $p(\mathcal{D})$ is the marginal likelihood that normalizes the distribution. For neural networks, the marginal likelihood $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$ is intractable. The integral requires summing over all possible parameter configurations, but neural networks produce complex nonlinear transformations through multiple layers that create multi-modal posterior geometries. Even simple networks exhibit weight symmetries—permuting hidden units produces identical functions—that create multiple posterior modes. These computational barriers motivate approximate inference methods.

56.3.2 Variational Inference for Bayesian Neural Networks

Variational inference for Bayesian Neural Networks (BNN) has been covered in detail in Chapter 12, including the evidence lower bound (ELBO), mean-field approximations, and the reparameterization trick for gradient computation [174]. In the context of reinforcement learning, variational inference is applied to Q-networks or policy networks by maintaining variational distributions $q_\phi(\theta)$ over network parameters θ [1397]. During training, the ELBO balances fitting observed temporal difference targets (the expected log-likelihood term) against staying close to the prior (the KL regularization term), resulting in posterior distributions that quantify epistemic uncertainty about value functions or policies. This uncertainty can then be leveraged for exploration through Thompson sampling (sampling one parameter configuration per episode) or upper confidence bounds (adding uncertainty bonuses to mean predictions), enabling principled exploration in high-dimensional state spaces where exact Bayesian inference would be intractable.

56.3.3 Variational Bayesian Deep Q-Networks

Variational inference integrates naturally with Deep Q-Networks (DQN), combining neural function approximation with principled uncertainty quantification for exploration. The algorithm alternates between collecting experience through interaction with the environment and updating the variational posterior over network parameters.

Temporal Difference Learning with Distributional Parameters

Standard DQN uses temporal difference learning to update Q-value estimates. The update rule computes TD targets:

$$y_t = r_t + \gamma \max_{a'} Q_{\theta_{\text{target}}}(s_{t+1}, a'), \quad (56.25)$$

where r_t is the observed reward, γ is the discount factor, and θ_{target} represents target network parameters that stabilize learning. The network parameters θ are then updated to minimize the squared TD error $\mathbb{E}[(Q_\theta(s_t, a_t) - y_t)^2]$. Variational Bayesian DQN maintains distributions over parameters instead of point estimates. The expected log-likelihood term in the ELBO becomes:

$$\mathbb{E}_{q_\phi(\theta)} \left[\sum_{(s,a,y) \in \mathcal{D}} \log p(y|s, a, \theta) \right] \quad (56.26)$$

Assuming Gaussian observation noise with variance σ_n^2 , this is proportional to negative squared error:

$$-\frac{1}{2\sigma_n^2} \mathbb{E}_{\theta \sim q_\phi} \sum_{(s,a,y) \in \mathcal{D}} (Q_\theta(s, a) - y)^2 \quad (56.27)$$

The algorithm samples parameter configurations from q_ϕ , computes TD errors for each sample, and uses these errors to update both means and variances through the ELBO gradient.

Target Networks and Posterior Updates

DQN uses target networks to stabilize temporal difference learning. Similarly, Variational Bayesian DQN maintains target variational parameters ϕ_{target} that update slowly:

$$\phi_{\text{target}} \leftarrow \tau \phi + (1 - \tau) \phi_{\text{target}}, \quad (56.28)$$

where $\tau \in (0, 1)$ is a small constant (typically 0.001) that controls update speed. TD targets are computed using samples from the target distribution $q_{\phi_{\text{target}}}(\theta)$ rather than the current distribution $q_\phi(\theta)$, reducing correlation between predictions and targets. Experience replay buffers store transitions (s_t, a_t, r_t, s_{t+1}) and sampling mini-batches from this buffer provides the dataset $\mathcal{D}$ for variational updates. The KL term $\text{KL}(q_\phi(\theta) \| p(\theta))$ acts as regularization, pulling the posterior toward the prior and preventing excessive variance growth. The uncertainty estimates from variational Bayesian DQN face systematic issues due to the bootstrapping nature of temporal difference learning. Because TD targets $y_t = r_t + \gamma \max_{a'} Q_{\theta_{\text{target}}}(s_{t+1}, a')$ are computed using the current Q-function estimate rather than unbiased Monte Carlo returns, the likelihood term in the ELBO becomes:

$$\mathbb{E}_{q_\phi(\theta)} \left[\sum_{(s,a,y) \in \mathcal{D}} \log p(y|s, a, \theta) \right] \approx -\frac{1}{2\sigma_n^2} \mathbb{E}_{\theta \sim q_\phi} \sum_{(s,a) \in \mathcal{D}} (Q_\theta(s, a) - \tilde{y})^2 \quad (56.29)$$

where $\tilde{y}$ itself depends on uncertain Q-value estimates. This recursive dependency means the posterior captures not only epistemic uncertainty about the true Q-function, but also propagates errors and biases from bootstrapped targets, leading to poorly calibrated uncertainty estimates that may be overconfident in some regions and underconfident in others. Additionally, the relative weighting between the expected log-likelihood term and the KL regularization term $\text{KL}(q_\phi(\theta) \| p(\theta))$ remains an open practical question. While the ELBO objective theoretically specifies equal weighting, practitioners often scale the KL term by $\beta \in (0, 1]$ or by $1/N$ where N is the dataset size, treating it as a form of weight decay.

Exploration through Posterior Sampling

Thompson sampling for exploration becomes straightforward with distributional parameters. At the start of each episode, sample one parameter configuration $\theta^* \sim q_\phi(\theta)$ and act greedily with respect to Q_{θ^*} throughout that episode:

$$a_t = \arg \max_a Q_{\theta^*}(s_t, a) \quad (56.30)$$

Different parameter samples produce different action preferences, and sampling from regions with high posterior variance naturally encourages exploration of uncertain regions. This exploration mechanism requires no explicit

bonuses or ϵ -greedy randomization; uncertainty guides exploration implicitly through the distribution over Q-functions. Alternatively, upper confidence bound exploration computes:

$$a_t = \arg \max_a \left[\mathbb{E}_{\theta \sim q_\phi} [Q_\theta(s_t, a)] + \beta \sqrt{\text{Var}_{\theta \sim q_\phi} [Q_\theta(s_t, a)]} \right], \quad (56.31)$$

where β controls optimism about uncertain actions. Computing the mean and variance requires sampling multiple parameter configurations, but this overhead is typically small compared to environment interaction costs.

Bootstrap Ensembles and Diversity

Bootstrap ensembles train M networks $\{Q_{\theta_1}, \dots, Q_{\theta_M}\}$ on different subsets of experience. Each network m optimizes:

$$\theta_m = \arg \max_{\theta} \sum_{(s, a, y) \in \mathcal{D}_m} \log p(y|s, a, \theta), \quad (56.32)$$

where $\mathcal{D}_m$ is a bootstrap sample of the experience replay buffer, created by sampling transitions with replacement. Different bootstrap samples lead to different trained networks, and the diversity of network predictions approximates posterior uncertainty. In RL, bootstrapped DQN maintains separate Q-networks and assigns each transition to a random subset of networks. Each network receives a different view of the data, leading to different learned Q-functions. For exploration, the algorithm samples one network at the beginning of each episode and acts greedily with respect to that network's Q-values throughout the episode. This Thompson sampling approximation balances exploration and exploitation: uncertain state-action pairs exhibit high variance across ensemble members, and sampling different networks naturally explores these uncertain regions. The ensemble variance provides an uncertainty measure:

$$\text{Var}[Q(s, a)] = \frac{1}{M} \sum_{m=1}^M (Q_{\theta_m}(s, a) - \bar{Q}(s, a))^2 \quad (56.33)$$

where $\bar{Q}(s, a) = \frac{1}{M} \sum_{m=1}^M Q_{\theta_m}(s, a)$ is the ensemble mean. This variance decreases as networks agree, indicating well-explored regions, and increases where networks disagree, indicating uncertainty that warrants exploration.

56.3.4 Bayesian Successor Features for Exploration

Successor Uncertainties [836] combine successor features with Bayesian linear regression to enable principled exploration under uncertainty. The key computational advantage is that instead of maintaining uncertainty over Q-values for all state-action pairs, the method tracks uncertainty only over a d -dimensional reward parameter vector w . When $d \ll |S \times A|$, this decomposition dramatically reduces computational burden while maintaining the ability to quantify uncertainty for exploration. The method performs Bayesian regression on the reward function, assuming a linear-Gaussian model:

$$r(s, a) = \phi(s, a)^\top w + \eta, \quad (56.34)$$

where $\phi(s, a) \in \mathbb{R}^d$ represents state-action features, $w \in \mathbb{R}^d$ is the reward parameter vector under which we place uncertainty, and $\eta \sim \mathcal{N}(0, \sigma^2)$ represents observation noise. Successor features capture the expected discounted occupancy of features when following policy π :

$$\psi^\pi(s, a) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t) \mid s_0 = s, a_0 = a \right], \quad (56.35)$$

where $\gamma \in [0, 1)$ is the discount factor. These can be learned via temporal difference methods analogous to Q-learning, or computed exactly in tabular settings. The Bayesian update places a Gaussian prior $w \sim \mathcal{N}(\mu_0, \Sigma_0)$ over reward parameters and updates it through standard Bayesian linear regression as reward observations are collected, yielding posterior $p(w|\mathcal{D}) = \mathcal{N}(\mu_n, \Sigma_n)$ where the posterior covariance Σ_n decreases with more observations. Exploration proceeds via Thompson sampling: at each step, the agent samples $w^{(i)} \sim \mathcal{N}(\mu_n, \Sigma_n)$ and selects $a_t = \arg \max_a \psi^\pi(s_t, a)^\top w^{(i)}$. This naturally trades off exploitation and exploration, as uncertainty in Σ_n encourages visiting state-action pairs where the reward structure is poorly understood.

56.4 Bayesian RL Methods Comparison

Table 56.1 summarizes the main Bayesian RL method families discussed in this chapter. Exact tabular methods correspond to the posterior-sampling, Bayes-adaptive planning, and Bayesian exploration approaches introduced in the early sections, including PSRL, BAMCP, and BEB. Gaussian process methods refer to GP-SARSA, PILCO, and related model-based control approaches, while variational BNNs and neural ensembles summarize the scalable neural approximations discussed above. The Linear-Gaussian row captures methods such as Bayesian successor features, where uncertainty is maintained over a low-dimensional linear reward or value model. MCMC sampling is included as a reference point rather than as a method developed in detail here: it represents a sampling-based route to Bayesian posterior approximation, but its computational cost and mixing difficulties often make it less practical than variational or ensemble approximations in large-scale reinforcement learning. The table organizes methods by problem class, computational complexity, uncertainty quality, implementation prerequisites, and primary failure modes. Each method family is analyzed in terms of scaling properties, implementation constraints, and application domains.

Table 56.1: Bayesian RL Methods Comparison.

Method Family	Applicable Problem Class	Complexity and Scaling	Uncertainty Quality	Implement. Prerequisites	Primary Failure Modes	Example Applications
Exact Tabular	$ \mathcal{S} \mathcal{A} < 10^5$ discrete state-action pairs	$O(\mathcal{S} \mathcal{A})$ per episode	Optimal epistemic uncertainty if model class correct	Compatible priors, sufficient exploration	Model misspecification, curse of dimensionality	Multi-armed bandits, grid-world exploration, clinical trial design
Linear-Gaussian	$V(s) = \phi(s)^\top \theta$ with known features ϕ	$O(d^2)$ per update (rank-1)	Well-calibrated uncertainty, biased mean if nonlinear	Feature engineering, stationarity assumptions	Poor feature design, nonlinear value functions	Portfolio optimisation, Linear-Quadratic Regulator (LQR) control, linear bandits
Gaussian Process	Smooth functions, $n < 5000$ observations	$O(n^3)$ training cost	Well-calibrated epistemic uncertainty for smooth functions	Kernel selection, hyperparameter tuning	Poor kernel choice, scalability limits	Robot control, hyperparameter optimisation, continuous bandits
Neural Ensembles	Large-scale problems, no structural assumptions	M heads $\times$ base training cost	Approximates epistemic uncertainty, planning-dependent quality	Ensemble diversity techniques, computational infrastructure	Heads converge identically, overconfidence	Atari games, robotic exploration, large-state RL
Variational BNN	Large networks requiring uncertainty estimates	$2 \times$ base training cost	Underestimates epistemic uncertainty (mean-field limitation)	Variational inference skills, optimisation debugging	Posterior collapse, poor calibration	Complex robotic control, neural dialogue systems
MCMC Sampling	Small-medium networks, complex posteriors	$10\text{-}50 \times$ base cost	Asymptotically exact epistemic uncertainty if converged	Sampling expertise, convergence assessment	Slow convergence, poor mixing	Safety-critical control, Bayesian neural policies

56.5 Applications and case studies

Robotics. Data-efficient control. PILCO achieves cart-pole swing-up in ~ 20 trials using GP dynamics with $M \approx 200$ inducing points; policy gradients propagate through predictive variances [397]. Actuator saturation can violate Gaussian moment matching and degrade rollouts [657]. Model-based alternatives plan under uncertainty via ensembles with MPC (PETS) [305] and GP-based probabilistic MPC with chance constraints [863]. These frameworks illustrate how Bayesian models can explicitly balance exploitation and active learning by planning trajectories that reduce uncertainty in regions critical for control performance.

Dialogue systems. Belief-aware planning under partial observability. Partially Observable Markov Decision Process (POMDP) managers maintain posteriors over goals and slots; Bayes-adaptive planning lowers failure rates relative to heuristic exploration [1332, 1423]. Monte Carlo planning with belief sampling scales to larger domains while retaining uncertainty-awareness [1515]. These approaches demonstrate how maintaining posterior beliefs over latent user intentions enables adaptive dialogue policies that query or confirm information when uncertainty is high.

Atari. Exploration via approximate posteriors. Bootstrapped DQN with randomised priors samples a head per episode to induce temporally extended exploration in sparse-reward tasks (e.g., *Montezuma's Revenge*) [1269, 1270]. Without priors, ensemble heads can collapse to similar Q-functions; excessive intrinsic rewards can reduce task returns [221]. The approach highlights how posterior sampling approximations in deep networks can provide

temporally coherent exploration signals in environments with sparse or delayed feedback.

Spatial navigation. *Structured inductive biases.* Ring-attractor modules maintain distributions over headings; embedding them in RL agents yields uncertainty-aware localisation and planning under partial observability [1230]. This line of work connects Bayesian reasoning with biologically inspired neural mechanisms, showing how structured representations can encode uncertainty in spatial variables to support robust navigation.

Safe control and constraint handling. *Safety during learning.* Bayesian model-based controllers plan over posterior dynamics with risk constraints. GP-based safe exploration uses confidence sets to restrict actions to high-probability safe regions [1644]. Surveyed mechanisms include Bayesian constraint models, risk-sensitive objectives, and shielded exploration [576]. Together, these methods formalize safety as a probabilistic constraint satisfaction problem, allowing exploration only when predicted violation probabilities remain below specified thresholds.

Recommenders and advertising (contextual bandits). *Personalisation with uncertainty estimates.* Posterior-sampling bandit methods adapt decisions using uncertainty over reward models; scalable deep approximations (variational BNNs, ensembles) perform well on large contextual problems [1397]. In this setting, uncertainty quantification drives exploration toward under-sampled users or contexts, improving long-term reward while controlling the cost of suboptimal recommendations.

56.6 Future directions

Two main open problems in scalable Bayesian RL are posterior approximation quality and safety guarantee provision. Current posterior approximations exhibit systematic biases that limit their effectiveness: variational methods underestimate uncertainty, leading to overconfident policies [543, 1274], while ensemble methods experience head collapse, where ensemble members converge to similar functions, and lack theoretical regret bounds [1270]. Structured variational families that explicitly capture temporal correlations across time steps [57] and ensemble architectures with diversity regularization to prevent collapse [310, 1739] represent approaches to address these limitations. The second problem concerns using uncertainty estimates to provide safety guarantees for real-world deployment: GP-based methods provide rigorous confidence bounds through posterior variance but require $O(n^3)$ computation that limits scalability [1644], while Deep Learning (DL) methods scale efficiently but produce poorly calibrated uncertainty estimates [1274]. Conformal prediction provides distribution-free safety bounds without distributional assumptions [1602], and Bayesian constraint modelling enables explicit incorporation of safety requirements [576], but integrating these techniques with deep RL at scale remains an active research direction.

Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Hany Abdulsamad, Sahel Iqbal, Christian A. Naesseth, Takuo Matsubara, and Adrien Corenflos. Maximin robust Bayesian experimental design, 2026.
- [3] Taiga Abe, Estefany Kelly Buchanan, Geoff Pleiss, Richard Zemel, and John Cunningham. Deep ensembles work, but are they necessary? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 33646–33660. Curran Associates, Inc., 2022.
- [4] J. Abellán, G.J. Klir, and S. Moral. Disaggregated total uncertainty measure for credal sets. *International Journal of General Systems*, 35(1):29–44, 2006.
- [5] Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. Parameters vs FLOPs: Scaling laws for optimal sparsity for mixture-of-experts language models, 2025.
- [6] Steven Adriaensen, Herilalaina Rakotoarison, Samuel Müller, and Frank Hutter. Efficient Bayesian learning curve extrapolation using prior-data fitted networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 19858–19886. Curran Associates, Inc., 2023.
- [7] S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance Sampling: Intrinsic Dimension and Computational Cost. *Statistical Science*, 32(3):405 – 431, 2017.
- [8] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4699–4711. Curran Associates, Inc., 2021.
- [9] Devanshu Agrawal, Theodore Papamarkou, and Jacob Hinkle. Wide neural networks with bottlenecks are deep Gaussian processes. *Journal of Machine Learning Research*, 21(175):1–66, 2020.
- [10] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 221–231, 2019.
- [11] Eirikur Agustsson and Lucas Theis. Universally quantized neural compression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12367–12376. Curran Associates, Inc., 2020.
- [12] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In

- I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [13] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.
- [14] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.
- [15] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [16] Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 130–140. PMLR, 2021.
- [17] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.
- [18] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P. Xing. Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 18(82):1–37, 2017.
- [19] Francesca Albertini, Eduardo D Sontag, and Vincent Maillot. Uniqueness of weights for neural networks. *Artificial Neural Networks for Speech and Vision*, 1993.
- [20] Angelo Alessandri, Patrizia Bagnerini, Mauro Gaggero, and Luca Mantelli. Parameter estimation of fire propagation models using level set methods. *Applied Mathematical Modelling*, 92:731–747, 2021.
- [21] Antonios Alexos, Alex J Boyd, and Stephan Mandt. Structured stochastic gradient MCMC. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 414–434. PMLR, 2022.
- [22] Elizabeth S. Allman, Catherine Matias, and John A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A):3099 – 3132, 2009.
- [23] Pierre Alquier. Approximate Bayesian inference. *Entropy*, 22(11), 2020.
- [24] Pierre Alquier. User-friendly introduction to PAC-Bayes bounds. *Foundations and Trends in Machine Learning*, 17(2):174–303, 2024.
- [25] Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of Gibbs posteriors. *Journal of Machine Learning Research*, 17(236):1–41, 2016.
- [26] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.
- [27] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer, 2005.
- [28] Kweku K Amfoh, Richard F Shaw, and George E Bonney. The use of logistic models for the analysis of codon frequencies of DNA sequences in terms of explanatory variables. *Biometrics*, 50(4):1054–1063, 1994.
- [29] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14927–14937. Curran Associates, Inc., 2020.
- [30] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

- [31] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7436–7447. Curran Associates, Inc., 2021.
- [32] Ning An, Huitong Ding, Jiaoyun Yang, Rhoda Au, and Ting F.A. Ang. Deep ensemble learning for alzheimer’s disease classification. *Journal of Biomedical Informatics*, 105:103411, 2020.
- [33] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.
- [34] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
- [35] Yashas Annadani, Nick Pawlowski, Joel Jennings, Stefan Bauer, Cheng Zhang, and Wenbo Gong. Bayesdag: Gradient-based posterior inference for causal discovery. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 1738–1763. Curran Associates, Inc., 2023.
- [36] Javier Antorán, Shreyas Padhy, Riccardo Barbano, Eric Nalisnick, David Janz, and José Miguel Hernández-Lobato. Sampling-based inference for large linear models, with application to linearised Laplace, 2022.
- [37] Diogo Leitao, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. Human-ai collaboration in decision-making: Beyond learning to defer, 2022.
- [38] Ziqiao Ao and Jinglai Li. On estimating the gradient of the expected information gain in Bayesian experimental design. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 20311–20319. AAAI Press, 2024.
- [39] Miki Aoyagi. Consideration on the learning efficiency of multiple-layered neural networks with linear units. *Neural Networks*, 172:106132, 2024.
- [40] Miki Aoyagi. Singular leaning coefficients and efficiency in learning theory, 2025.
- [41] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.
- [42] Ienkaran Arasaratnam, Simon Haykin, and Robert J. Elliott. Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007.
- [43] Julyan Arbel, Konstantinos Pitas, Mariia Vladimirova, and Vincent Fortuin. A Primer on Bayesian Neural Networks: Review and Debates. *Statistical Science*, 41(2):316 – 353, 2026.
- [44] Michael Arbel, David Salinas, and Frank Hutter. Equitabpfn: A target-permutation equivariant prior fitted networks, 2025.
- [45] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W. Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks, 2018.
- [46] S. Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.
- [47] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.
- [48] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3): 337–404, 1950.

- [49] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [50] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.
- [51] Vyrion Arvanitis, Angelos Aslanidis, Emanuel Sommer, and David Rügamer. bde: A python package for Bayesian deep ensembles via mile, 2026.
- [52] Jordan Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. Gone fishing: Neural active learning with Fisher embeddings. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8927–8939. Curran Associates, Inc., 2021.
- [53] Jordan T. Ash and Ryan P. Adams. On warm-starting neural network training, 2019.
- [54] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [55] Kianoosh Ashouritaklimi and Tom Rainforth. Active learning with task-driven representations for messy pools, 2025.
- [56] Kianoosh Ashouritaklimi, Valentin Kilian, Daolang Huang, Tom Rainforth, and François Caron. Prediction-powered active testing, 2026.
- [57] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020.
- [58] Roy Assaf and Anika Schumann. Explainable deep neural networks for multivariate time series predictions. In *IJCAI*, pages 6488–6490. Macao, 2019.
- [59] Anthony C Atkinson, A N Donev, and R D Tobias. *Optimum Experimental Designs, with SAS*. Oxford University Press, 2007.
- [60] Les Atlas, David Cohn, and Richard Ladner. Training connectionist networks with queries and selective sampling. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.
- [61] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.
- [62] Thomas Augustin, Frank P. A. Coolen, Gert de Cooman, and Matthias C. M. Troffaes. *Introduction to imprecise probabilities*. John Wiley & Sons,, West Sussex, England, 2014.
- [63] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
- [64] Tim Austin. Exchangeable Random Arrays. In *Notes for IAS workshop*, 2012.
- [65] Grace Avecilla, Julie N. Chuong, Fangfei Li, Gavin Sherlock, David Gresham, and Yoav Ram. Neural networks enable efficient and accurate simulation-based inference of evolutionary parameters from adaptation dynamics. *PLOS Biology*, 20(5):1–32, 2022.
- [66] Nihat Ay, Jürgen Jost, Hông Vân Lê, and Lorenz Schwachhöfer. *Information Geometry*, volume 64 of *Ergebnisse der Mathematik und ihrer Grenzgebiete 34*. Springer International Publishing, Cham, 2017.

- [67] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.
- [68] Jimmy Ba, Murat A. Erdogdu, Marzyeh Ghassemi, Shengyang Sun, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Understanding the variance collapse of SVGD in high dimensions. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [69] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- [70] Joris Baan, Nico Daheim, Evgenia Ilia, Dennis Ulmer, Haau-Sing Li, Raquel Fernández, Barbara Plank, Rico Sennrich, Chrysoula Zerva, and Wilker Aziz. Uncertainty in natural language generation: From theory to applications, 2023.
- [71] Francis Bach. Active learning for misspecified generalized linear models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.
- [72] François Bachoc and Agnès Lagnoux. Posterior contraction rates for constrained deep Gaussian processes in density estimation and classification. *Communications in Statistics - Theory and Methods*, 54(3):774–811, 2025.
- [73] Bae, Noh, and Sutherland. Generalized coverage for more robust low-budget active learning. *European Conference on Computer Vision*, 2024.
- [74] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.
- [75] Yu-Ting Bai, Wei Jia, Xue-Bo Jin, Ting-Li Su, Jian-Lei Kong, and Zhi-Gang Shi. Nonstationary time series prediction based on deep echo state network tuned by Bayesian optimization. *Mathematics*, 11(6), 2023.
- [76] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.
- [77] Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, 29(3):599–615, 2019.
- [78] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient monte-carlo Bayesian optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21524–21538. Curran Associates, Inc., 2020.
- [79] Krishnakumar Balasubramanian, Sayan Banerjee, and Anna Korba. Uniform-in-time propagation-of-chaos for Stein variational gradient descent, 2026.
- [80] Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C. Cresswell, and Rahul G. Krishnan. Causalpfn: Amortized causal effect estimation via in-context learning, 2025.
- [81] Tara Baldacchino, Elizabeth J. Cross, Keith Worden, and Jennifer Rowson. Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, 66-67:178–200, 2016.
- [82] Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Alessandro Ingrosso, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences*, 113(48): E7655–E7662, 2016.
- [83] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality, 2016.

- [84] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.
- [85] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [86] Johannes Ballé, Nick Johnston, and David Minnen. Integer networks for data compression with latent-variable models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [87] Johannes Ballé, Sung Jin Hwang, and Eirikur Agustsson. TensorFlow Compression: Learned data compression, 2022. Software available from <http://github.com/tensorflow/compression>.
- [88] Jona Ballé, Philip A. Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353, 2021.
- [89] Robert Bamler. Understanding entropy coding with asymmetric numeral systems (ANS): a statistician’s perspective, 2022.
- [90] Sayan Banerjee, Krishnakumar Balasubramanian, and Promit Ghosal. Improved finite-particle convergence rates for Stein variational gradient descent, 2024.
- [91] Kirk C. Bansak, Elisabeth Paulson, and Dominik Rothenhaeusler. Learning under random distributional shifts. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 3943–3951. PMLR, 2024.
- [92] Vikas Bansal and Christina Boucher. Sequencing technologies and analyses: Where have we been and where are we going? *iScience*, 18:37–41, 2019. RECOMB-Seq 2019.
- [93] David Barber and Felix Agakov. The IM algorithm: a variational approach to information maximization. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.
- [94] Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2):816 – 845, 2023.
- [95] Maria Maddalena Barbieri and James O. Berger. Optimal predictive model selection. *The Annals of Statistics*, 32(3):870 – 897, 2004.
- [96] Nicola Barileto, Bernardo Flores, and Stephen G. Walker. Posterior consistency in parametric models via a tighter notion of identifiability, 2025.
- [97] Nicola Barileto, Huy Nguyen, Nhat Ho, and Alessandro Rinaldo. On Bayesian softmax-gated mixture-of-experts models, 2026.
- [98] Yasir Zubayr Barlas, Sabina J. Sloman, and Samuel Kaski. Robust experimental design via generalised Bayesian inference, 2025.
- [99] David Barrado, Paul Mollière, Polychronis Patapis, Michiel Min, Pascal Tremblin, Francisco Ardevol Martinez, Niall Whiteford, Malavika Vasist, Ioannis Argyriou, Matthias Samland, Pierre-Olivier Lagage, Leen Decin, Rens Waters, Thomas Henning, María Morales-Calderón, Manuel Guedel, Bart Vandenbussche, Olivier Absil, Pierre Baudoz, Anthony Boccaletti, Jeroen Bouwman, Christophe Cossou, Alain Coulais, Nicolas Crouzet, René Gastaud, Alistair Glasse, Adrian M. Glauser, Inga Kamp, Sarah Kendrew, Oliver Krause, Fred Lahuis, Michael Mueller, Göran Olofsson, John Pye, Daniel Rouan, Pierre Royer, Silvia Scheithauer, Ingo Waldmann, Luis Colina, Ewine F. van Dishoeck, Tom Ray, Göran Östlin, and Gillian Wright. 15nh3 in the atmosphere of a cool brown dwarf. *Nature*, 624(7991):263–266, 2023.
- [100] Andrew Barron, Mark J. Schervish, and Larry Wasserman. The consistency of posterior distributions in nonparametric problems. *The Annals of Statistics*, 27(2):536 – 561, 1999.

- [101] David J Bartholomew, Martin Knott, and Irini Moustaki. *Latent variable models and factor analysis: A unified approach*. John Wiley & Sons, third edition edition, 2011.
- [102] Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008.
- [103] Arved Bartuska, Luis Espath, and Raúl Tempone. Laplace-based strategies for Bayesian optimal experimental design with nuisance uncertainty. *Statistics and Computing*, 35(1):12, 2025. doi: 10.1007/s11222-024-10544-z. URL <https://doi.org/10.1007/s11222-024-10544-z>.
- [104] Eric Baum and Kevin Lang. Constructing hidden units using examples and queries. In R.P. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann, 1990.
- [105] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.
- [106] K. Bayse, M. Lejter, and Keiji Kanazawa. Reducing uncertainty in navigation and exploration. In *Conference on Uncertainty in Artificial Intelligence*, 1990.
- [107] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.
- [108] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. CompressAI: a PyTorch library and evaluation platform for end-to-end compression research, 2020.
- [109] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.
- [110] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.
- [111] B.M. Bell and F.W. Cathey. The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993.
- [112] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.
- [113] Alexis Bellot, Kim Branson, and Mihaela van der Schaar. Neural graphical modelling in continuous-time: consistency guarantees and algorithms, 2021.
- [114] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.
- [115] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.
- [116] Hakim Benkirane, Yoann Pradat, Stefan Michiels, and Paul-Henry Cournède. Customics: A versatile deep-learning based strategy for multi-omics integration. *PLOS Computational Biology*, 19(3):1–19, 2023.
- [117] C.H. Bennett, P.W. Shor, J.A. Smolin, and A.V. Thapliyal. Entanglement-assisted capacity of a quantum channel and the reverse shannon theorem. *IEEE Transactions on Information Theory*, 48(10):2637–2655, 2002.
- [118] W. R. Bennett. Spectra of quantized signals. *The Bell System Technical Journal*, 27(3):446–472, 1948.
- [119] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. Learning invariances in neural networks from training data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17605–17616. Curran Associates, Inc., 2020.

- [120] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks, 2020.
- [121] Federico Bergamin, Pablo Moreno-Muñoz, Søren Hauberg, and Georgios Arvanitidis. Riemannian Laplace approximations for Bayesian neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 31066–31095. Curran Associates, Inc., 2023.
- [122] James O. Berger. The robust Bayesian viewpoint. In J. B. Kadane, editor, *Robustness of Bayesian Analysis*, pages 63–124. North-Holland, Amsterdam, 1984.
- [123] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 1985.
- [124] Toby Berger. Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*, 2003.
- [125] Richard Bergna, Stefan Depeweg, and José Miguel Hernández-Lobato. Decoupled pfn: Identifiable epistemic-aleatoric decomposition via structured synthetic priors, 2026.
- [126] Robert H. Berk. Consistency a Posteriori. *The Annals of Mathematical Statistics*, 41(3):894 – 906, 1970.
- [127] L Mark Berliner. Hierarchical Bayesian time series models. In *Maximum entropy and Bayesian methods: Santa Fe, New Mexico, USA, 1995 Proceedings of the fifteenth international workshop on maximum entropy and bayesian methods*, pages 15–22. Springer, 1996.
- [128] Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, 2004.
- [129] Jose M. Bernardo. Expected Information as Expected Utility. *The Annals of Statistics*, 7(3):686 – 690, 1979.
- [130] Jose M. Bernardo. Reference posterior distributions for Bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):113–128, 1979.
- [131] Espen Bernton, Pierre E. Jacob, Mathieu Gerber, and Christian P. Robert. Approximate Bayesian computation with the Wasserstein distance. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(2):235–269, 2019.
- [132] Gabrielle Berrada, Jannik Kossen, Freddie Bickford Smith, Muhammed Razzak, Yarin Gal, and Thomas Rainforth. Scaling up active testing to large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 173670–173697. Curran Associates, Inc., 2025.
- [133] P. Berti, E. Regazzini, and P. Rigo. Well calibrated, coherent forecasting systems. *Theory of Probability & Its Applications*, 42(1):82–102, 1998.
- [134] Patrizia Berti, Luca Pratelli, and Pietro Rigo. Limit theorems for a class of identically distributed random variables, 2004.
- [135] Patrizia Berti, Luca Pratelli, and Pietro Rigo. Exchangeable sequences driven by an absolutely continuous random measure. *The Annals of Probability*, 41(3):2090–2102, 2013.
- [136] Patrizia Berti, Emanuela Dreassi, Luca Pratelli, and Pietro Rigo. A predictive approach to Bayesian nonparametrics. 2019.
- [137] Patrizia Berti, Emanuela Dreassi, Fabrizio Leisen, Pietro Rigo, and Luca Pratelli. Bayesian predictive inference without a prior, 2021.
- [138] Patrizia Berti, Emanuela Dreassi, Fabrizio Leisen, Pietro Rigo, and Luca Pratelli. A probabilistic view on predictive constructions for Bayesian learning, 2022.
- [139] Carlo Berzuini and Walter Gilks. *RESAMPLE-MOVE Filtering with Cross-Model Jumps*, pages 117–138. Springer New York, New York, NY, 2001.
- [140] Alexandros Beskos, Natesh Pillai, Gareth Roberts, Jesus-Maria Sanz-Serna, and Andrew Stuart. Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501 – 1534, 2013.

- [141] Alexandros Beskos, Dan Crisan, and Ajay Jasra. On the stability of sequential Monte Carlo methods in high dimensions. *The Annals of Applied Probability*, 24(4):1396 – 1445, 2014.
- [142] Alexandros Beskos, Ajay Jasra, Kody Law, Raul Tempone, and Yan Zhou. Multilevel sequential Monte Carlo samplers. *Stochastic Processes and their Applications*, 127(5):1417–1440, 2017.
- [143] Alexandros Beskos, Ajay Jasra, Kody Law, Youssef Marzouk, and Yan Zhou. Multilevel sequential Monte Carlo with dimension-independent likelihood-informed proposals. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):762–786, 2018.
- [144] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo, 2017.
- [145] Michael Betancourt. Scalable Bayesian inference with Hamiltonian Monte Carlo. YouTube, 2018. Accessed: February 8, 2026.
- [146] Michael Betancourt and Mark Girolami. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79(30), 2015.
- [147] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In Andreea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 49–56. ACM, 2009.
- [148] Uddipta Bhardwaj, James Alvey, Benjamin Kurt Miller, Samaya Nissanke, and Christoph Weniger. Sequential simulation-based inference for gravitational wave signals. *Physical Review D*, 108(4):042004, 2023.
- [149] Manu Bhat, Jonghyun Park, Jianke Yang, Nima Dehmamy, Robin Walters, and Rose Yu. Atlasd: Automatic local symmetry discovery, 2025.
- [150] Shrijita Bhattacharya and Tapabrata Maiti. Statistical foundation of variational Bayes neural networks. *Neural Networks*, 137:151–173, 2021.
- [151] Shrijita Bhattacharya, Zihuan Liu, and Tapabrata Maiti. Variational Bayes neural network: Posterior consistency, classification accuracy and computational challenges, 2020.
- [152] Sohom Bhattacharya, Jianqing Fan, and Debarghya Mukherjee. Deep neural networks for nonparametric interaction models with diverging dimension. *The Annals of Statistics*, 52(6):2738 – 2766, 2024.
- [153] Adel Bibi, Modar Alfadly, and Bernard Ghanem. Analytic expressions for probabilistic moments of PL-DNN with Gaussian input. In *Computer Vision and Pattern Recognition*, 2018.
- [154] Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-oriented Bayesian active learning. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 7331–7348. PMLR, 2023.
- [155] Freddie Bickford Smith, Adam Foster, and Tom Rainforth. Making better use of unlabelled data in Bayesian active learning. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 847–855. PMLR, 2024.
- [156] Freddie Bickford Smith, Jannik Kossen, Eleanor Trollope, Mark Van Der Wilk, Adam Foster, and Tom Rainforth. Rethinking aleatoric and epistemic uncertainty. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4345–4359. PMLR, 2025.
- [157] Patrick Billingsley. *Probability and Measure*. Wiley Series in Probability and Mathematical Statistics. Wiley, New York, third edition edition, 1995.
- [158] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

- [159] Christopher M. Bishop. Mixture density networks. Copyright © 1994, Christopher M. Bishop. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (<https://creativecommons.org/licenses/by-nc-nd/4.0/>), 1994.
- [160] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [161] Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2023.
- [162] Christopher M. Bishop and Markus Svensén. Bayesian hierarchical mixtures of experts. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, pages 57–64. Morgan Kaufmann, 2003.
- [163] Devansh Bisla, Jing Wang, and Anna Choromanska. Low-pass filtering SGD for recovering flat optima in the deep learning optimization landscape. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8299–8339. PMLR, 2022.
- [164] P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016.
- [165] Blackwell. Comparison of experiments. *Berkeley Symposium on Mathematical Statistics and Probability*, 1951.
- [166] David A. Blackwell and M. A. Girshick. *Theory of Games and Statistical Decisions*. Wiley, 1954.
- [167] R. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, 18(4):460–473, 1972.
- [168] Tom Blau, Edwin V. Bonilla, Iadine Chades, and Amir Dezfouli. Optimizing sequential experimental design with deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2107–2128. PMLR, 2022.
- [169] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [170] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [171] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 950–959. PMLR, 2020.
- [172] Michael G. B. Blum and Olivier François. Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20(1):63–73, 2010.
- [173] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam’s razor. *Information Processing Letters*, 24(6):377–380, 1987.
- [174] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 2015. PMLR.
- [175] Natalia Bochkina. Bernstein–von mises theorem and misspecified models: A review. In Denis Belomestny, Cristina Butucea, Enno Mammen, Eric Moulines, Markus Reiß, and Vladimir V. Ulyanov, editors, *Foundations of Modern Statistics*, pages 355–380, Cham, 2023. Springer International Publishing.
- [176] S. Bochner. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, 108(1):378–410, 1933.
- [177] Jan Boelts, Jan-Matthis Lueckmann, Richard Gao, and Jakob H Macke. Flexible and efficient simulation-based inference for models of decision-making. *eLife*, 11:e77220, 2022.

- [178] Jan Boelts, Michael Deistler, Manuel Gloeckler, Álvaro Tejero-Cantero, Jan-Matthis Lueckmann, Guy Moss, Peter Steinbach, Thomas Moreau, Fabio Muratore, Julia Linhart, Conor Durkan, Julius Vetter, Benjamin Kurt Miller, Maternus Herold, Abolfazl Ziaemehr, Matthijs Pals, Theo Gruner, Sebastian Bischoff, Nastya Krouglova, Richard Gao, Janne K. Lappalainen, Bálint Mucsányi, Felix Pei, Auguste Schulz, Zinovia Stefanidi, Pedro Rodrigues, Cornelius Schröder, Faried Abu Zaid, Jonas Beck, Jaivardhan Kapoor, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. sbi reloaded: a toolkit for simulation-based inference workflows, 2024.
- [179] Nicholas M Boffi and Eric Vanden-Eijnden. Probability flow solution of the fokker–planck equation. *Machine Learning: Science and Technology*, 4(3):035012, 2023.
- [180] Vladimir I. Bogachev. *Gaussian Measures*. American Mathematical Society, 1998.
- [181] Vladimir Igorevich Bogachev. *Measure theory*. Springer, 2007.
- [182] Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (WMT18). In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névél, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels, 2018. Association for Computational Linguistics.
- [183] Annabel Bolek, Marcel Croon, and Jacques Hagenaars. Estimating latent structure models with categorical variables: One-step versus three-step estimators. *Political Analysis*, 12(1):3–27, 2004.
- [184] Matthew Bonas and Stefano Castruccio. Calibration of spatio-temporal forecasts from citizen science urban air pollution data with sparse recurrent neural networks, 2021.
- [185] Matthew Bonas, Christopher K. Wikle, and Stefano Castruccio. Calibrated forecasts of quasi-periodic climate processes with deep echo state networks and penalized quantile regression. *Environmetrics*, 35(1): e2833, 2024.
- [186] Edwin Bonilla, Kian Chai, and Christopher Williams. Multi-task Gaussian process prediction. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- [187] Georges Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. In *Annales des Télécommunications*, volume 19, pages 203–220. Springer, 1964.
- [188] Gregory P. Bopp, Benjamin A. Shaby, and Raphaël Huser. A hierarchical max-infinitely divisible spatial model for extreme precipitation. *Journal of the American Statistical Association*, 116(533):93–106, 2021.
- [189] Siegfried Böś and Manfred Opper. Dynamics of training. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.
- [190] Thijs Bos and Johannes Schmidt-Hieber. Convergence rates of deep ReLU networks for multiclass classification. *Electronic Journal of Statistics*, 16(1):2724 – 2773, 2022.
- [191] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 557–565. PMLR, 2017.
- [192] Leon Bottou. Stochastic gradient learning in neural networks. In *Proceedings of the International Conference on Neural Networks (Neuro-Nîmes 91)*, Nîmes, France, 1991.
- [193] Léon Bottou. *On-line learning and stochastic approximations*, pages 9–42. Cambridge University Press, USA, 1999.
- [194] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMP-STAT’2010*, pages 177–186. Springer, 2010.
- [195] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

- [196] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
- [197] Kouroche Bouchiat, Alexander Immer, Hugo Yèche, Gunnar Ratsch, and Vincent Fortuin. Improving neural additive models with Bayesian principles. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4416–4443. PMLR, 2024.
- [198] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 1951.
- [199] George E. P. Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.
- [200] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [201] Jonathan R Bradley, Shijie Zhou, and Xu Liu. Deep hierarchical generalized transformation models for spatio-temporal data with discrepancy errors. *Spatial Statistics*, 55:100749, 2023.
- [202] John Bradshaw, Alexander G. de G. Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks, 2017.
- [203] Mark Braverman and Ankit Garg. Public vs private coin in bounded-round information. In *International Colloquium on Automata, Languages, and Programming*, pages 502–513. Springer, 2014.
- [204] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape, 2019.
- [205] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [206] Leo Breiman. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231, 2001.
- [207] David R Brillinger, Haiganoush K Preisler, and John W Benoit. Risk assessment: a forest fire example. *Lecture Notes-Monograph Series*, pages 177–196, 2003.
- [208] Christopher Brislawn. Traceable integral kernels on countably generated measure spaces. *Pacific Journal of Mathematics*, 150(2):229–240, 1991.
- [209] Andrey G. Bronevich and Igor N. Rozenberg. Generalized hartley measures on credal sets. In Andrés Cano, Jasper De Bock, Enrique Miranda, and Serafin Moral, editors, *Proceedings of the Twelfth International Symposium on Imprecise Probability: Theories and Applications*, volume 147 of *Proceedings of Machine Learning Research*, pages 32–41. PMLR, 2021.
- [210] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.
- [211] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.
- [212] Tim Brown, Steve Leach, Brent Wachter, and Billy Gardunio. The extreme 2018 northern california fire season. *Bulletin of the American Meteorological Society*, 101(1):S1 – S4, 2020.
- [213] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

- [214] Paul Brunzema, Mikkel Jordahn, John Willes, Sebastian Trimpe, Jasper Snoek, and James Harrison. Variational last layers for Bayesian optimization. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*, 2024.
- [215] Alexander Buchholz, Nicolas Chopin, and Pierre E. Jacob. Adaptive Tuning of Hamiltonian Monte Carlo Within Sequential Monte Carlo. *Bayesian Analysis*, 16(3):745 – 771, 2021.
- [216] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [217] Thang Bui, Daniel Hernandez-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep Gaussian processes for regression using approximate expectation propagation. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1472–1481, New York, New York, USA, 2016. PMLR.
- [218] Thang D Bui, Cuong Nguyen, and Richard Turner. Streaming sparse Gaussian process approximations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [219] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
- [220] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.
- [221] Yuri Burda, Harri Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [222] Michael Craig Burkhardt and Kouros Modarresi. Digital experience enhancement using an ensemble deep learning model, 2023. US Patent 11,816,562.
- [223] David R. Burt, Carl Edward Rasmussen, and Mark van der Wilk. Variational orthogonal features, 2020.
- [224] Alberto Cabezas, Adrien Corenflos, Junpeng Lao, Rémi Louf, Antoine Carnec, Kaustubh Chaudhari, Reuben Cohn-Gordon, Jeremie Coullon, Wei Deng, Sam Duffield, Gerardo Durán-Martín, Marcin Elan-tkowski, Dan Foreman-Mackey, Michele Gregori, Carlos Iguaran, Ravin Kumar, Martin Lysy, Kevin Murphy, Juan Camilo Orduz, Karm Patel, Xi Wang, and Rob Zinkov. Blackjax: Composable Bayesian inference in jax, 2024.
- [225] Yuhang Cai and Lek-Heng Lim. Distances between probability distributions of different dimensions. *IEEE Transactions on Information Theory*, 68(6):4020–4031, 2022.
- [226] Roberto Calandra, Jan Peters, Carl E Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016.
- [227] Jake Callahan and Tommie Catanach. On the misinformation in a statistical experiment. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.
- [228] Jake Callahan, Andrew Chin, Jason Pacheco, and Tommie Catanach. Reverse-annealed sequential Monte Carlo for efficient Bayesian optimal experiment design. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 64910–64932. Curran Associates, Inc., 2025.
- [229] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 28266–28279. Curran Associates, Inc., 2022.
- [230] Patrick Cannon, Daniel Ward, and Sebastian M. Schmon. Investigating the impact of model misspecification in neural simulation-based inference, 2022.

- [231] Wenhan Cao, Tianyi Zhang, Zeju Sun, Chang Liu, Stephen S. T. Yau, and Shengbo Eben Li. Nonlinear Bayesian filtering with natural gradient Gaussian approximation, 2024.
- [232] Francesco Caporali, Stefano Favaro, and Dario Trevisan. Student-t processes as infinite-width limits of posterior Bayesian neural networks, 2025.
- [233] Michele Caprio. Refined Pinsker’s and reverse Pinsker’s inequalities for probability distributions of different dimensions. *IEEE Access*, 10:116425–116431, 2022.
- [234] Michele Caprio and Sayan Mukherjee. Finite admixture models: a bridge with stochastic geometry and choquet theory, 2020.
- [235] Michele Caprio and Sayan Mukherjee. Ergodic theorems for dynamic imprecise probability kinematics. *International Journal of Approximate Reasoning*, 152:325–343, 2023.
- [236] Michele Caprio, Souradeep Dutta, Kuk Jin Jang, Vivian Lin, Radoslav Ivanov, Oleg Sokolsky, and Insup Lee. Credal Bayesian deep learning. *Transactions on Machine Learning Research*, 2024.
- [237] Michele Caprio, Yusuf Sale, Eyke Hüllermeier, and Insup Lee. A Novel Bayes’ Theorem for Upper Probabilities. In Fabio Cuzzolin and Maryam Sultana, editors, *Epistemic Uncertainty in Artificial Intelligence*, pages 1–12, Cham, 2024. Springer Nature Switzerland.
- [238] Michele Caprio, David Stutz, Shuo Li, and Arnaud Doucet. Conformalized credal regions for classification with ambiguous ground truth, 2024.
- [239] Michele Caprio, Shireen K. Manchingal, and Fabio Cuzzolin. Credal and interval deep evidential classifications, 2025.
- [240] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017.
- [241] José Carrillo and Jakub Skrzeczkowski. Convergence and stability results for the particle system in the Stein gradient descent method. *Mathematics of Computation*, 94(354):1793–1814, 2025.
- [242] Arran Carter and Torben Sell. Partial trace-class Bayesian neural networks, 2025.
- [243] Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. Handling sparsity via the horseshoe. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 73–80, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.
- [244] Ismaël Castillo and Paul Egels. Posterior and variational inference for deep neural networks with heavy-tailed weights, 2024.
- [245] Ismaël Castillo and Thibault Randrianarisoa. Deep horseshoe Gaussian processes, 2024.
- [246] Olivier Catoni. Pac-Bayesian supervised classification: The thermodynamics of statistical learning, 2007.
- [247] Daniel R. Cavagnaro, Jay I. Myung, Mark A. Pitt, and Janne V. Kujala. Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science. *Neural Computation*, 22(4):887–905, 2010.
- [248] Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(70):2079–2107, 2010.
- [249] Center for AI Safety, Scale AI & HLE Contributors Consortium. A benchmark of expert-level academic questions to assess ai capabilities. *Nature*, 649(8099):1139–1146, 2026.
- [250] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

- [251] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In Marcello Federico, Sebastian Stüker, and François Yvon, editors, *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17, 2014.
- [252] Neil K. Chada, Ajay Jasra, Kody J. H. Law, and Sumeetpal S. Singh. Bayesian deep learning with multilevel trace-class neural networks. *SIAM Journal on Mathematics of Data Science*, 7(3):1210–1240, 2025.
- [253] Sunrit Chakraborty, Rayleigh Lei, and XuanLong Nguyen. Learning topic hierarchies by tree-directed latent variable models, 2024.
- [254] Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273 – 304, 1995.
- [255] Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [256] Noirit Kiran Chandra, Antonio Canale, and David B. Dunson. Escaping the curse of dimensionality in Bayesian model-based clustering. *Journal of Machine Learning Research*, 24(144):1–42, 2023.
- [257] Paul E. Chang, Prakhar Verma, S.T. John, Arno Solin, and Mohammad Emtiyaz Khan. Memory-based dual Gaussian processes for sequential learning. In *International Conference on Machine Learning*, 2023.
- [258] Peter G. Chang, Kevin Patrick Murphy, and Matt Jones. On diagonal approximations to the extended Kalman filter for online training of Bayesian neural networks. In *Continual Lifelong Learning Workshop at ACML 2022*, 2022.
- [259] Peter G. Chang, Gerardo Durán-Martín, Alex Shestopaloff, Matt Jones, and Kevin Patrick Murphy. Low-rank extended Kalman filtering for online learning of neural networks from streaming data. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup, editors, *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pages 1025–1071. PMLR, 2023.
- [260] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1356–1367. Curran Associates, Inc., 2020.
- [261] Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.
- [262] Sotirios P. Chatzis and Yiannis Demiris. Echo state Gaussian process. *IEEE Transactions on Neural Networks*, 22(9):1435–1445, 2011.
- [263] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.
- [264] Tatjana Chavdarova, Gauthier Gidel, François Fleuret, and Simon Lacoste-Julien. Reducing noise in GAN training with variance reduced extragradient. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [265] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5(6):910–927, 1993.
- [266] Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

- [267] Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. Self-supervised learning for few-shot image classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1745–1749. IEEE, 2021.
- [268] George H. Chen. Deep kernel survival analysis and subject-specific survival time prediction intervals. In Finale Doshi-Velez, Jim Fackler, Ken Jung, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 5th Machine Learning for Healthcare Conference*, volume 126 of *Proceedings of Machine Learning Research*, pages 537–565. PMLR, 2020.
- [269] Lei Chen, Xiaohui Zhong, Hao Li, Jie Wu, Bo Lu, Deliang Chen, Shang-Ping Xie, Libo Wu, Qingchen Chao, Chensen Lin, et al. A machine learning model that outperforms conventional global subseasonal forecast models. *Nature Communications*, 15(1):6425, 2024.
- [270] Peng Chen and Omar Ghattas. Projected Stein variational gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1947–1958. Curran Associates, Inc., 2020.
- [271] Ricky T. Q. Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [272] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [273] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Beijing, China, 2014. PMLR.
- [274] Tianyi Chen, Yingzhou Lu, Nan Hao, Yuanyuan Zhang, Capucine Van Rechem, Jintai Chen, and Tianfan Fu. Uncertainty quantification on clinical trial outcome prediction, 2024.
- [275] Wenlin Chen, Austin Tripp, and José Miguel Hernández-Lobato. Meta-learning adaptive deep kernel Gaussian processes for molecular property prediction. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [276] Wenlin Chen, Mingtian Zhang, Jiajun He, Zijing Ou, José Miguel Hernández-Lobato, Bernhard Schölkopf, and David Barber. Diffratio: Training one-step diffusion models without teacher supervision, 2025.
- [277] Wenlong Chen and Yingzhen Li. Calibrating transformers via sparse Gaussian processes. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [278] Wenlong Chen, Naoki Kiyohara, Harrison Zhu, Jacob Curran-Sebastian, Samir Bhatt, and Yingzhen Li. Recurrent memory for online interdomain Gaussian processes. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 134836–134873. Curran Associates, Inc., 2025.
- [279] Wenlong Chen, Bolian Li, Ruqi Zhang, and Yingzhen Li. Bayesian computation in deep learning, 2025.
- [280] Yanzhi Chen, Dinghuai Zhang, Michael Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models, 2020.
- [281] Yanzhi Chen, Michael U. Gutmann, and Adrian Weller. Is learning summary statistics necessary for likelihood-free inference? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 4529–4544. PMLR, 2023.
- [282] Yifan Chen, Daniel Zhengyu Huang, Jiaoyang Huang, Sebastian Reich, and Andrew M. Stuart. Gradient flows for sampling: Mean-field models, Gaussian approximations and affine invariance, 2023.

- [283] Yifan Chen, Daniel Zhengyu Huang, Jiaoyang Huang, Sebastian Reich, and Andrew M Stuart. Sampling via gradient flows in the space of probability measures, 2023.
- [284] Yutian Chen, Abram L Friesen, Feryal Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas. Modular meta-learning with shrinkage. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2858–2869. Curran Associates, Inc., 2020.
- [285] Ching-An Cheng and Byron Boots. Variational inference for Gaussian process models with linear complexity. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [286] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024.
- [287] Xiang Cheng, Niladri S. Chatterji, Peter L. Bartlett, and Michael I. Jordan. Underdamped Langevin MCMC: A non-asymptotic analysis. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 300–323. PMLR, 2018.
- [288] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized Gaussian mixture likelihoods and attention modules, 2020.
- [289] Badr-Eddine Chérif-Abdellatif, Pierre Alquier, and Mohammad Emtiyaz Khan. A generalization bound for online variational inference. In Wee Sun Lee and Taiji Suzuki, editors, *Proceedings of The Eleventh Asian Conference on Machine Learning*, volume 101 of *Proceedings of Machine Learning Research*, pages 662–677. PMLR, 2019.
- [290] Herman Chernoff. Sequential Design of Experiments. *The Annals of Mathematical Statistics*, 30(3):755 – 770, 1959.
- [291] Herman Chernoff. *Sequential Analysis and Optimal Design*. Society for Industrial and Applied Mathematics, 1972.
- [292] Sinho Chewi. *Log-Concave Sampling*. Online, 2025.
- [293] Sinho Chewi, Thibaut Le Gouic, Chen Lu, Tyler Maunu, and Philippe Rigollet. SVGD as a kernelized Wasserstein gradient flow of the chi-squared divergence. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2098–2109. Curran Associates, Inc., 2020.
- [294] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.
- [295] Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 2552–2573. PMLR, 2022.
- [296] Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.
- [297] Nicolas Chopin and Omiros Papaspiliopoulos. *An introduction to sequential Monte Carlo*. Springer, 2020.
- [298] P.A. Chou, T. Lookabaugh, and R.M. Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–42, 1989.
- [299] Kamal Choudhary, Brian DeCost, Chi Chen, Anubhav Jain, Francesca Tavazza, Ryan Cohn, Cheol Woo Park, Alok Choudhary, Ankit Agrawal, Simon J. L. Billinge, Elizabeth Holm, Shyue Ping Ong, and Chris Wolverton. Recent advances and applications of deep learning methods in materials science. *npj Computational Materials*, 8(1):59, 2022.

- [300] Deepro Choudhury, Sinead Williamson, Adam Golinski, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent information gathering with LLMs and Bayesian experimental design. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 84375–84405, 2026.
- [301] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.
- [302] Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254, 1957.
- [303] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [304] Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’11, pages 195–203, New York, NY, USA, 2011. Association for Computing Machinery.
- [305] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [306] Hwan Chung, Brian P. Flaherty, and Joseph L. Schafer. Latent class logistic regression: application to marijuana use and attitudes among high school seniors. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 169(4):723–743, 2006.
- [307] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2606–2615, New York, New York, USA, 2016. PMLR.
- [308] Tristan Cinquin, Alexander Immer, Max Horn, and Vincent Fortuin. Pathologies in priors and inference for Bayesian transformers, 2021.
- [309] Tristan Cinquin, Marvin Pförtner, Vincent Fortuin, Philipp Hennig, and Robert Bamler. Fsp-Laplace: Function-space priors for the Laplace approximation in Bayesian deep learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 13897–13926. Curran Associates, Inc., 2024.
- [310] Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard E. Turner. Conservative uncertainty estimation by fitting prior networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [311] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George Bm Van Den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc’Aurelio Ranzato, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4057–4086. PMLR, 2022.
- [312] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.
- [313] Ross M. Clarke, Elre Talea Oldewage, and José Miguel Hernández-Lobato. Scalable one-pass optimisation of high-dimensional weight-update hyperparameters by implicit differentiation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

- [314] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.
- [315] Adam D. Cobb and Brian Jalaian. Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 675–685. PMLR, 2021.
- [316] Adam D. Cobb, Michael D. Himes, Frank Soboczenski, Simone Zorzan, Molly D. O’Beirne, Atılım Güneş Baydin, Yarin Gal, Shawn D. Domagal-Goldman, Giada N. Arney, Daniel Angerhausen, and 2018 NASA FDL Astrobiology Team II. An ensemble of Bayesian neural networks for exoplanetary atmospheric retrieval. *The Astronomical Journal*, 158(1):33, 2019.
- [317] Jon Cockayne, Chris J. Oates, Ilse C.F. Ipsen, and Mark Girolami. A Bayesian Conjugate Gradient Method (with Discussion). *Bayesian Analysis*, 14(3):937 – 1012, 2019.
- [318] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 2016. PMLR.
- [319] Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [320] David Cohn. Neural network exploration using optimal experiment design. In J. Cowan, G. Tesauro, and J. Alsppector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.
- [321] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- [322] David Cohn, Zoubin Ghahramani, and Michael Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.
- [323] David Cohn, Zoubin Ghahramani, and Michael Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 1996.
- [324] Alex Cole, Benjamin K. Miller, Samuel J. Witte, Maxwell X. Cai, Meiert W. Grootes, Francesco Nattino, and Christoph Weniger. Fast and credible likelihood-free cosmology with truncated marginal neural ratio estimation. *Journal of Cosmology and Astroparticle Physics*, 2022(09):004, 2022.
- [325] Basile Confavreux, Poornima Ramesh, Pedro J Goncalves, Jakob H Macke, and Tim Vogels. Meta-learning families of plasticity rules in recurrent spiking networks using simulation-based inference. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 13545–13558. Curran Associates, Inc., 2023.
- [326] Bai Cong, Nico Daheim, Yuesong Shen, Rio Yokota, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Improving LoRA with variational learning, 2025.
- [327] R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977.
- [328] Samantha R Cook, Andrew Gelman, and Donald B Rubin. Validation of software for Bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, 2006.
- [329] F.P.A. Coolen. *Imprecise highest density regions related to intervals of measures*. Memorandum COSOR. Technische Universiteit Eindhoven, 1992.
- [330] Thomas E. Coons and Xun Huan. A multifidelity estimator of the expected information gain for Bayesian optimal experimental design. *SIAM/ASA Journal on Uncertainty Quantification*, 13(4):1990–2021, 2025.

- [331] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020.
- [332] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International conference on algorithmic learning theory*, pages 67–82. Springer, 2016.
- [333] Nathaël Da Costa, Bálint Mucsányi, and Philipp Hennig. Geometric Gaussian approximations of probability distributions, 2025.
- [334] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [335] Jarrad Courts, Adrian Wills, Thomas Schön, and Brett Ninness. Variational system identification for nonlinear state-space models, 2020.
- [336] Jarrad Courts, Adrian Wills, and Thomas B. Schön. Gaussian variational state estimation for nonlinear state-space models, 2020.
- [337] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2006.
- [338] R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.
- [339] Richard T Cox. *Algebra of probable inference*. The John Hopkins Press, Baltimore, MD, 1961.
- [340] Diane Coyle and Lucy Hampton. 21st century progress in computing. *Telecommunications Policy*, 48(1):102649, 2024.
- [341] Fabio Gagliardi Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.
- [342] Harald Cramér. *Mathematical methods of statistics*. Princeton University Press, revised edition edition, 1999.
- [343] Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers, 2015.
- [344] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.
- [345] Noel Cressie and Christopher K. Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2011.
- [346] Dan Crisan, Joaquín Míguez, and Gonzalo Ríos-Muñoz. On the performance of parallelisation schemes for particle filtering. *EURASIP Journal on Advances in Signal Processing*, 2018:1–18, 2018.
- [347] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.
- [348] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2018.
- [349] Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, 21(8):1470–1480, 2024.
- [350] Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep Gaussian processes. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 884–893. PMLR, 2017.

- [351] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [352] Giuseppe Da Prato and Jerzy Zabczyk. *Stochastic equations in infinite dimensions*. Cambridge university press, 2014.
- [353] Nico Daheim, Thomas Möllenhoff, Edoardo M. Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 11836–11854, 2024.
- [354] Nico Daheim, Clara Meister, Thomas Möllenhoff, and Iryna Gurevych. Uncertainty-aware decoding with minimum Bayes risk. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 7212–7234, 2025.
- [355] Chenguang Dai, Jeremy Heng, Pierre E. Jacob, and Nick Whiteley. An invitation to sequential Monte Carlo samplers. *Journal of the American Statistical Association*, 117(539):1587–1600, 2022.
- [356] Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics.
- [357] Zhenwen Dai, Andreas Damianou, James Hensman, and Neil Lawrence. Gaussian process models with parallelization and GPU acceleration, 2014.
- [358] Zhenwen Dai, Andreas Damianou, Javier González, and Neil Lawrence. Variational auto-encoded deep Gaussian processes, 2015.
- [359] Arnak S. Dalalyan and Avetik Karagulyan. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019.
- [360] Andreas Damianou. *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, University of Sheffield, 2015.
- [361] Andreas Damianou and Neil D. Lawrence. Deep Gaussian processes. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA, 2013. PMLR.
- [362] Andreas Damianou and Neil D. Lawrence. Semi-described and semi-supervised learning with Gaussian processes, 2015.
- [363] Andreas Damianou, Michalis Titsias, and Neil Lawrence. Variational Gaussian process dynamical systems. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.
- [364] Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research*, 17(42):1–62, 2016.
- [365] Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [366] Felix Dangel, Runa Eschenhagen, Weronika Ormaniec, Andres Fernandez, Lukas Tatzel, and Agustinus Kristiadi. Position: Curvature matrices should be democratized via linear operators, 2025.
- [367] Felix Dangel, Bálint Mucsányi, Tobias Weber, and Runa Eschenhagen. Kronecker-factored approximate curvature (kfac) from scratch, 2025.

- [368] Francesco D’Angelo, Vincent Fortuin, and Florian Wenzel. On Stein variational neural network ensembles, 2021.
- [369] Michael J. Daniels, Antonio Linero, and Jason Roy. *Bayesian Nonparametrics for Causal Inference and Missing Data*. Chapman and Hall/CRC, 2023.
- [370] Tivadar Danka and Peter Horvath. modal: A modular active learning framework for python, 2018.
- [371] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [372] Jacob E. Darling and Kyle J. DeMars. Minimization of the Kullback–Leibler divergence for nonlinear estimation. *Journal of Guidance, Control, and Dynamics*, 40(7):1739–1748, 2017.
- [373] Sanjoy Dasgupta and Daniel J. Hsu. Hierarchical sampling for active learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 208–215. ACM, 2008.
- [374] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
- [375] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press, second edition edition, 1984.
- [376] Dawid. Coherent measures of discrepancy, uncertainty and dependence, with applications to Bayesian predictive experimental design. Technical report, University College London, 1998.
- [377] A. P. Dawid. Present position and potential developments: Some personal views statistical theory: the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.
- [378] Maximilian Dax, Stephen R. Green, Jonathan Gair, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time gravitational wave science with neural posterior estimation. *Phys. Rev. Lett.*, 127: 241103, 2021.
- [379] Maximilian Dax, Stephen R. Green, Jonathan Gair, Nihar Gupte, Michael Pürrer, Vivien Raymond, Jonas Wildberger, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time inference for binary neutron star mergers using machine learning. *Nature*, 639(8053):49–53, 2025.
- [380] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless Bayesian deep learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20089–20103. Curran Associates, Inc., 2021.
- [381] Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antoran, and Jose Miguel Hernandez-Lobato. Bayesian deep learning via subnetwork inference. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2510–2521. PMLR, 2021.
- [382] C. Mitchell Dayton and George B. Macready. Concomitant-variable latent-class models. *Journal of the American Statistical Association*, 83(401):173–178, 1988.
- [383] Bruno de Finetti. La prévision: ses lois logiques, ses sources subjectives. In *Annales de l’Institut Henri Poincaré*, volume 7, pages 1–68. Presses Universitaires de France, 1937.
- [384] Bruno de Finetti. *Theory of Probability*, volume 1. John Wiley & Sons, 1974.
- [385] Bruno de Finetti. *Theory of Probability*, volume 2. John Wiley & Sons, 1975.
- [386] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

- [387] Denis Pereira de Lima, Rafael Francisco Viana Sanches, and Emerson Carlos Pedrino. Neural network training using unscented and extended Kalman filter. *Robotics and Automation Engineering Journal*, 1(4), 2017.
- [388] Daniel Augusto de Souza, Alexander Nikitin, ST John, Magnus Ross, Mauricio A Álvarez, Marc Deisenroth, João Paulo Gomes, Diego Mesquita, and César Lincoln Mattos. Thin and deep Gaussian processes. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 13138–13148. Curran Associates, Inc., 2023.
- [389] Peter Dear. *Discipline and Experience: the Mathematical Way in the Scientific Revolution*. University of Chicago Press, 1995.
- [390] Aimée M. Deaton, Shaun Webb, Alastair R.W. Kerr, Robert S. Illingworth, Jacky Guy, Robert Andrews, and Adrian Bird. Cell type-specific DNA methylation at intragenic cpg islands in the immune system. *Genome Research*, 21(7):1074–1086, 2011.
- [391] DeepSeek-AI. Deepseek LLM: Scaling open-source language models with longtermism, 2024.
- [392] DeepSeek-AI. Deepseek-v3 technical report, 2024.
- [393] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025.
- [394] DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai, Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang, Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao, Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao, Chong Ruan, Fuli Luo, and Wenfeng Liang. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence, 2024.
- [395] DeGroot. *Optimal Statistical Decisions*. John Wiley & Sons, 1970.
- [396] M. H. DeGroot. Uncertainty, Information, and Sequential Experiments. *The Annals of Mathematical Statistics*, 33(2):404 – 419, 1962.
- [397] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 465–472. Omnipress, 2011.
- [398] Michael Deistler, Pedro J Goncalves, and Jakob H Macke. Truncated proposals for scalable and hassle-free simulation-based inference. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 23135–23149. Curran Associates, Inc., 2022.
- [399] Michael Deistler, Jakob H. Macke, and Pedro J. Gonçalves. Energy-efficient network activity from disparate circuit parameters. *Proceedings of the National Academy of Sciences*, 119(44):e2207632119, 2022.
- [400] Michael Deistler, Jan Boelts, Peter Steinbach, Guy Moss, Thomas Moreau, Manuel Gloeckler, Pedro L. C. Rodrigues, Julia Linhart, Janne K. Lappalainen, Benjamin Kurt Miller, Pedro J. Gonçalves, Jan-Matthis Lueckmann, Cornelius Schröder, and Jakob H. Macke. Simulation-based inference: A practical guide, 2025.
- [401] Pierre Del Moral. *Feynman-Kac formulae: Genealogical and interacting particle systems with applications*. Springer, 2004.
- [402] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.
- [403] Sylvain Delattre and Nicolas Fournier. On the Kozachenko–Leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.
- [404] Arnaud Delaunoy, Joeri Hermans, François Rozet, Antoine Wehenkel, and Gilles Louppe. Towards reliable simulation-based inference with balanced neural ratio estimation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20025–20037. Curran Associates, Inc., 2022.

- [405] Arnaud Delaunoy, Benjamin Kurt Miller, Patrick Forré, Christoph Weniger, and Gilles Louppe. Balancing simulation-based inference for conservative posteriors, 2023.
- [406] Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94 – 128, 1999.
- [407] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- [408] Felix den Breejen and Se-Young Yun. Attic: A new architecture for tabular in-context learning transformers, 2025.
- [409] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [410] Zhijie Deng, Peng Cui, and Jun Zhu. Towards accelerated model training via Bayesian data selection. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8513–8527. Curran Associates, Inc., 2023.
- [411] Misha Denil, Babak Shakibi, Laurent Dinh, Marc’Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning, 2013.
- [412] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield. Large automatic learning, rule extraction, and generalization. *Complex Systems*, 1(5):877–922, 1987.
- [413] Thierry Dencœux. An evidential neural network model for regression based on random fuzzy numbers. In Sylvie Le Hégarat-Masclé, Isabelle Bloch, and Emanuel Aldea, editors, *Belief Functions: Theory and Applications*, pages 57–66, Cham, 2022. Springer International Publishing.
- [414] Thierry Dencœux. Quantifying prediction uncertainty in regression using random fuzzy sets: The ennreg model. *IEEE Transactions on Fuzzy Systems*, 31(10):3690–3699, 2023.
- [415] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1184–1193. PMLR, 2018.
- [416] Rahul S. Desikan, Florent Ségonne, Bruce Fischl, Brian T. Quinn, Bradford C. Dickerson, Deborah Blacker, Randy L. Buckner, Anders M. Dale, R. Paul Maguire, Bradley T. Hyman, Marilyn S. Albert, and Ronald J. Killiany. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage*, 31(3):968–980, 2006.
- [417] Gianluca Detommaso, Tiangang Cui, Youssef Marzouk, Alessio Spantini, and Robert Scheichl. A Stein variational Newton method. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [418] Gianluca Detommaso, Alberto Gasparin, Michele Donini, Matthias Seeger, Andrew Gordon Wilson, and Cedric Archambeau. Fortuna: A library for uncertainty quantification in deep learning. *Journal of Machine Learning Research*, 25(238):1–7, 2024.
- [419] Holger Dette. Designing experiments with respect to ‘standardized’ optimality criteria. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):97–110, 1997.
- [420] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, 1986.
- [421] Rayen Dhahri, Alexander Immer, Bertrand Charpentier, Stephan Günnemann, and Vincent Fortuin. Shaving weights with Occam’s razor: Bayesian sparsification for neural networks using the marginal likelihood. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 24959–24989. Curran Associates, Inc., 2024.
- [422] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

- [423] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022.
- [424] Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. Avoiding latent variable collapse with generative skip models. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2397–2405. PMLR, 2019.
- [425] Nghiem Tuong Diep, Huy Nguyen, Chau Nguyen, Minh Le, Duy Minh Ho Nguyen, Daniel Sonntag, Mathias Niepert, and Nhat Ho. On zero-initialized attention: Optimal prompt and gating factor estimation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 13713–13745. PMLR, 2025.
- [426] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [427] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow distributions, 2017.
- [428] Fangyu Ding, Ding Ding, Sijin Chen, Kaibo Wang, Peng Xu, Zijin Feng, Haoli Bai, Kai Han, Youliang Yan, Binhang Yuan, and Jiacheng Sun. Beyond masks: Efficient, flexible diffusion language models via deletion-insertion processes. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 103093–103149, 2026.
- [429] Jiarui Ding, Xian Adiconis, Sean K. Simmons, Monika S. Kowalczyk, Cynthia C. Hession, Nemanja D. Marjanovic, Travis K. Hughes, Marc H. Wadsworth, Tyler Burks, Lan T. Nguyen, John Y. H. Kwon, Boaz Barak, William Ge, Amanda J. Kedaigle, Shaina Carroll, Shuqiang Li, Nir Hacohen, Orit Rozenblatt-Rosen, Alex K. Shalek, Alexandra-Chloé Villani, Aviv Regev, and Joshua Z. Levin. Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. *Nature Biotechnology*, 38(6):737–746, 2020.
- [430] Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D. Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [431] Wisdom Mdumiseni Dlamini. Application of Bayesian networks for fire risk mapping using gis and remote sensing data. *GeoJournal*, 76:283–296, 2011.
- [432] Bao Gia Doan, Afshar Shamsi, Xiao-Yu Guo, Arash Mohammadi, Hamid Alinejad-Rokny, Dino Sejdinovic, Damien Teney, Damith C. Ranasinghe, and Ehsan Abbasnejad. Bayesian low-rank learning (bella): A practical approach to Bayesian neural networks. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 16298–16307. AAAI Press, 2025.
- [433] N.A.K. Doan, W. Polifke, and L. Magri. Physics-informed echo state networks. *Journal of Computational Science*, 47:101237, 2020.
- [434] T. J. Dodwell, C. Ketelsen, R. Scheichl, and A. L. Teckentrup. A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1075–1108, 2015.
- [435] Daniel Dold, David Ruegamer, Beate Sick, and Oliver Dürr. Bayesian semi-structured subspace inference. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1819–1827. PMLR, 2024.
- [436] Daniel Dold, Julius Kobialka, Nicolai Palm, Emanuel Sommer, David Rügamer, and Oliver Dürr. Paths and ambient spaces in neural loss landscapes. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 10–18. PMLR, 2025.

- [437] Daniel Dold, Emanuel Sommer, Julius Kobialka, Oliver Dürr, and David Rügamer. On the construction and implications of low-loss valleys in LoRA-based Bayesian inference, 2026.
- [438] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.
- [439] Runpei Dong, Zhanhong Tan, Mengdi Wu, Linfeng Zhang, and Kaisheng Ma. Finding the task-optimal low-bit sub-distribution in deep neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5343–5359. PMLR, 2022.
- [440] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.
- [441] M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.
- [442] Joseph L Doob. Application of the theory of martingales. *Le calcul des probabilités et ses applications*, pages 23–27, 1949.
- [443] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddhartha V Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 2403–2426. Curran Associates, Inc., 2023.
- [444] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017.
- [445] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.
- [446] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In Dan Crisan and Boris Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.
- [447] Arnaud Doucet, Nando De Freitas, and Neil James Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer, 2001.
- [448] Matthew Dowling, Yuan Zhao, and Il Memming Park. exponential family dynamical systems (xfads): Large-scale nonlinear Gaussian state-space modeling. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 13458–13488. Curran Associates, Inc., 2024.
- [449] K. Doya. Bifurcations in the learning of recurrent neural networks. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, volume 6, pages 2777–2780 vol.6, 1992.
- [450] Simon J. D.Prince. *Understanding Deep Learning*. MIT Press, 2023.
- [451] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR, 2018.
- [452] Drugman, Pytköinen, and Kneser. Active and semi-supervised learning in ASR: benefits on the acoustic and language models. In *Interspeech*, 2016.
- [453] Francesco D' Angelo and Vincent Fortuin. Repulsive deep ensembles are Bayesian. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3451–3465. Curran Associates, Inc., 2021.
- [454] Xinting Du and Hejin Wang. Efficient estimation of expected information gain in Bayesian experimental design with multi-index Monte Carlo. *Statistics and Computing*, 34(6), 2024.

- [455] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [456] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.
- [457] Didier Dubois and Eyke Hüllermeier. Comparing probability measures using possibility theory: A notion of relative peakedness. *International Journal of Approximate Reasoning*, 45(2):364–385, 2007. Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005).
- [458] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.
- [459] Jarek Duda. Asymmetric numeral systems, 2009.
- [460] Michael O. G. Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive MDPs*. PhD thesis, University of Massachusetts Amherst, 2002.
- [461] Samuel Duffield, Kaelan Donatella, Johnathan Chiu, Phoebe Klett, and Daniel Simpson. Scalable Bayesian learning with posteriors. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 11935–11958, 2025.
- [462] Andrew Duncan, Nikolas Nüsken, and Lukasz Szpruch. On the geometry of Stein variational gradient descent. *Journal of Machine Learning Research*, 24(56):1–39, 2023.
- [463] Matthew M. Dunlop, Mark A. Girolami, Andrew M. Stuart, and Aretha L. Teckentrup. How deep are deep Gaussian processes? *Journal of Machine Learning Research*, 19(54):1–46, 2018.
- [464] David B. Dunson, Natesh Pillai, and Ju-Hyun Park. Bayesian density regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):163–183, 2007.
- [465] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations, 2021.
- [466] Gerardo Duran-Martin, Aleyna Kara, and Kevin Murphy. Efficient online Bayesian inference for neural bandits. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 6002–6021. PMLR, 2022.
- [467] Gerardo Duran-Martin, Matias Altamirano, Alex Shestopaloff, Leandro Sánchez-Betancourt, Jeremias Knoblauch, Matt Jones, Francois-Xavier Briol, and Kevin Patrick Murphy. Outlier-robust Kalman filtering through generalised Bayes. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12138–12171. PMLR, 2024.
- [468] Gerardo Duran-Martin, Leandro Sánchez-Betancourt, Álvaro Cartea, and Kevin Murphy. Martingale posterior neural networks for fast sequential decision making, 2025.
- [469] Nikita Durasov, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Masksembles for uncertainty estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13539–13548, 2021.
- [470] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [471] Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2771–2781. PMLR, 2020.
- [472] Alain Durmus, Szymon Majewski, and Błażej Miasojedow. Analysis of Langevin Monte Carlo via convex optimization. *Journal of Machine Learning Research*, 20(73):1–46, 2019.

- [473] Nicolas Durrande, David Ginsbourger, and Olivier Roustant. Additive kernels for Gaussian process modeling, 2011.
- [474] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2782–2792. PMLR, 2020.
- [475] Sergey Dushenko, Kapildeb Ambal, and Robert D. McMichael. Sequential Bayesian experiment design for optically detected magnetic resonance of nitrogen-vacancy centers. *Phys. Rev. Appl.*, 14:054036, 2020.
- [476] Vincent Dutordoir, Nicolas Durrande, and James Hensman. Sparse Gaussian processes with spherical harmonic features. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2793–2802. PMLR, 2020.
- [477] Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande. Deep neural networks as point estimates for deep Gaussian processes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9443–9455. Curran Associates, Inc., 2021.
- [478] Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P. Deisenroth, and ST John. Gpflux: A library for deep Gaussian processes, 2021.
- [479] David Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive Gaussian processes, 2011.
- [480] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1166–1174, Atlanta, Georgia, USA, 2013. PMLR.
- [481] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 202–210, Reykjavik, Iceland, 2014. PMLR.
- [482] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [483] Gintare Karolina Dziugaite and Daniel Roy. Entropy-SGD optimizes the prior of a PAC-Bayes bound: Generalization properties of entropy-SGD and data-dependent priors. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1377–1386. PMLR, 2018.
- [484] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [485] B. Efron. Why isn’t everyone a Bayesian? *The American Statistician*, 40(1):1–5, 1986.
- [486] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010.
- [487] Jennifer Stine Elam, Matthew F. Glasser, Michael P. Harms, Stamatios N. Sotiropoulos, Jesper L.R. Andersson, Gregory C. Burgess, Sandra W. Curtiss, Robert Oostenveld, Linda J. Larson-Prior, Jan-Mathijs Schoffelen, Michael R. Hodge, Eileen A. Cler, Daniel M. Marcus, Deanna M. Barch, Essa Yacoub, Stephen M. Smith, Kamil Ugurbil, and David C. Van Essen. The human connectome project: A retrospective. *NeuroImage*, 244:118543, 2021.
- [488] Daniel Ellsberg. Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics*, 75(4): 643–669, 1961.

- [489] Lasse Elsemüller, Hans Olischläger, Marvin Schmitt, Paul-Christian Bürkner, Ullrich Köthe, and Stefan T. Radev. Sensitivity-aware amortized Bayesian inference, 2023.
- [490] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks, 2017.
- [491] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey, 2018.
- [492] Andrew Engel, Zhichao Wang, Natalie S. Frank, Ioana Dumitriu, Sutanay Choudhury, Anand Sarwate, and Tony Chiang. Faithful and efficient explanations for neural networks via neural tangent kernel surrogate models, 2023.
- [493] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [494] Epoch AI. Data on machine learning hardware. <https://epoch.ai/data/machine-learning-hardware>, 2024.
- [495] Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. Graphpfn: A prior-data fitted graph foundation model, 2025.
- [496] Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models. 2026.
- [497] David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 2021.
- [498] Piero Esposito. BLiTz - Bayesian layers in torch zoo (a Bayesian deep learning library for torch). <https://github.com/piEsposito/blitz-bayesian-deep-learning/>, 2020.
- [499] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.
- [500] Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. LM-polygraph: Uncertainty estimation for language models. In Yansong Feng and Els Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 446–461, Singapore, 2023. Association for Computational Linguistics.
- [501] Fabian Falck, Ziyu Wang, and Christopher C. Holmes. Is in-context learning in large language models Bayesian? A Martingale perspective. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12784–12805. PMLR, 2024.
- [502] Maciej Falkiewicz, Naoya Takeishi, Imahn Shekhzadeh, Antoine Wehenkel, Arnaud Delaunoy, Gilles Louppe, and Alexandros Kalousis. Calibrating neural simulation-based inference with differentiable coverage probability. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 1082–1099. Curran Associates, Inc., 2023.
- [503] Josua Faller and Jörg Martin. Low rank based subspace inference for the Laplace approximation of Bayesian neural networks, 2025.
- [504] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.
- [505] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National Science Review*, 1(2): 293–314, 2014.

- [506] Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [507] Stefano Favaro, Sandra Fortini, and Stefano Peluchetti. Deep stable neural networks: Large-width asymptotics and convergence rates. *Bernoulli*, 29(3):2574 – 2597, 2023.
- [508] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: Semi-automatic abc, 2010.
- [509] Paul Fearnhead, Christopher Nemeth, Chris J. Oates, and Chris Sherlock. *Scalable Monte Carlo for Bayesian Learning*. Institute of Mathematical Statistics Monographs. Cambridge University Press, 2025.
- [510] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.
- [511] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- [512] Mohammed Fellaji and Frédéric Pennerath. The epistemic uncertainty hole: an issue of Bayesian neural networks, 2024.
- [513] Chi Feng. Optimal Bayesian experimental design in the presence of model error. Master’s thesis, Massachusetts Institute of Technology, 2015.
- [514] Thomas S. Ferguson. *Mathematical Statistics: a Decision Theoretic Approach*. Academic Press, 1967.
- [515] Martin Ferienc, Partha Maji, Matthew Mattina, and Miguel Rodrigues. On the effects of quantisation on model uncertainty in Bayesian neural networks. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 929–938. PMLR, 2021.
- [516] Patrick Fernandes, António Farinhas, Ricardo Rei, José G. C. de Souza, Perez Ogayo, Graham Neubig, and Andre Martins. Quality-aware decoding for neural machine translation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States, 2022. Association for Computational Linguistics.
- [517] Mara Finkelstein and Markus Freitag. Mbr and qe finetuning: Training-time distillation of the best and most expensive decoding methods. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 39960–39978, 2024.
- [518] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.
- [519] Gianluca Finocchio and Johannes Schmidt-Hieber. Posterior contraction for deep Gaussian process priors. *Journal of Machine Learning Research*, 24(66):1–49, 2023.
- [520] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3165–3176. PMLR, 2020.
- [521] Marc Finzi, Gregory Benton, and Andrew G Wilson. Residual pathway priors for soft equivariance constraints. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 30037–30049. Curran Associates, Inc., 2021.
- [522] Kirsten Fischer, Javed Lindner, David Dahmen, Zohar Ringel, Michael Krämer, and Moritz Helias. Critical feature learning in deep neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13660–13690. PMLR, 2024.

- [523] Bethany Fishbein, Pierre Foy, and Liqun Yin. TIMSS 2019 user guide for the international database. Technical report, TIMSS & PIRLS International Study Center, Lynch School of Education and Human Development, Boston College and International Association for the Evaluation of Educational Achievement (IEA), Chestnut Hill, MA, 2021.
- [524] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character*, 222 (594-604):309–368, 1922.
- [525] Ronald A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, 1925.
- [526] Ronald A. Fisher. The arrangement of field experiments. *Journal of the Ministry of Agriculture*, 1926.
- [527] Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, 1935.
- [528] Tor Erlend Fjelde, Kai Xu, David Widmann, Mohamed Tarek, Cameron Pfiffer, Martin Trapp, Seth D. Axen, Xianda Sun, Markus Hauru, Penelope Yong, Will Tebbutt, Zoubin Ghahramani, and Hong Ge. Turing.jl: A general-purpose probabilistic programming language. *ACM Trans. Probab. Mach. Learn.*, 1 (3), 2025.
- [529] Gergely Flamich. Greedy Poisson rejection sampling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 37089–37127. Curran Associates, Inc., 2023.
- [530] Gergely Flamich. *Data Compression with Relative Entropy Coding*. PhD thesis, Apollo - University of Cambridge Repository, 2024.
- [531] Gergely Flamich and Deniz Gündüz. Data compression with stochastic codes. *IEEE BITS the Information Theory Magazine*, pages 1–14, 2026.
- [532] Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. Compressing images by encoding their latent representations with relative entropy coding. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16131–16141. Curran Associates, Inc., 2020.
- [533] Gergely Flamich, Stratis Markou, and Jose Miguel Hernandez-Lobato. Fast relative entropy coding with a* coding. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6548–6577. PMLR, 2022.
- [534] Gergely Flamich, Sharang M Sriramu, and Aaron B Wagner. The redundancy of non-singular channel simulation. In *2025 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2025.
- [535] Emlyn Flint and E. Maré. Regime-based tactical allocation for equity factors and balanced portfolios. *South African Actuarial Journal*, 19(1):27–52, 2019.
- [536] Loic Le Folgoc, Vasileios Baltatzis, Sujal Desai, Anand Devaraj, Sam Ellis, Octavio E. Martinez Manzanera, Arjun Nair, Huaqi Qiu, Julia Schnabel, and Ben Glocker. Is mc dropout Bayesian?, 2021.
- [537] E Fong and C C Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 2020.
- [538] Edwin Fong. *The predictive view of Bayesian inference*. PhD thesis, University of Oxford, 2021.
- [539] Edwin Fong and Briec Lehmann. A predictive approach to Bayesian nonparametric survival analysis, 2022.
- [540] Edwin Fong, Chris Holmes, and Stephen G Walker. Martingale posterior distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(5):1357–1391, 2023.
- [541] Andrew Foong, Wessel Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8284–8295. Curran Associates, Inc., 2020.

- [542] Andrew Foong, David Burt, Yingzhen Li, and Richard Turner. On the expressiveness of approximate inference in Bayesian neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15897–15908. Curran Associates, Inc., 2020.
- [543] Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. Between the lines: A between-class Bayesian neural network approach to adversarial examples. In *International Conference on Learning Representations*, 2019.
- [544] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [545] Alexander I.J Forrester, András Söbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463 (2088):3251–3269, 2007.
- [546] Alex Forster, Desi R. Ivanova, and Tom Rainforth. Improving robustness to model misspecification in Bayesian experimental design. In *7th Symposium on Advances in Approximate Bayesian Inference – Workshop Track*, 2025.
- [547] Vincent Fortuin. Priors in Bayesian deep learning: A review. *International Statistical Review*, 90(3): 563–591, 2022.
- [548] Vincent Fortuin, Adrià Garriga-Alonso, Mark van der Wilk, and Laurence Aitchison. Bnnpriors: A library for Bayesian neural network inference with different prior distributions. *Software Impacts*, 9:100079, 2021.
- [549] Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Rätsch, Richard E. Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [550] Adam Foster, Martin Jankowiak, Elias Bingham, Paul Horsfall, Yee Whye Teh, Thomas Rainforth, and Noah Goodman. Variational Bayesian optimal experimental design. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [551] Adam Foster, Martin Jankowiak, Matthew O’Meara, Yee Whye Teh, and Tom Rainforth. A unified stochastic gradient approach to designing Bayesian-optimal experiments. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2959–2969. PMLR, 2020.
- [552] Adam Foster, Desi R Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential Bayesian experimental design. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3384–3395. PMLR, 2021.
- [553] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. TRADI: Tracking deep neural network weight distributions. In *European Conference on Computer Vision*, pages 105–121. Springer, 2020.
- [554] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [555] Allan Franklin. *The Neglect of Experiment*. Cambridge University Press, 1986.
- [556] Allan Franklin. *Experiment, Right or Wrong*. Cambridge University Press, 1990.
- [557] Peter I. Frazier. A tutorial on Bayesian optimization, 2018.
- [558] Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. High quality rather than high model probability: Minimum Bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825, 2022.

- [559] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2):133–168, 1997.
- [560] B. J. Frey and G. E. Hinton. Efficient stochastic source coding and an application to a Bayesian network source model. *The Computer Journal*, 40(2 and 3):157–165, 1997.
- [561] Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication*. The MIT Press, 1998.
- [562] K. Fukumizu and S. Watanabe. Error estimation and learning data arrangement for neural networks. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 2, pages 777–780 vol.2, 1994.
- [563] Kenji Fukumizu. Active learning in multilayer perceptrons. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.
- [564] Jørund Gåsemyr and Aliaksandr Hubin. Prior distributions expressing ignorance about convex increasing failure rates. *Scandinavian Journal of Statistics*, 50(2):452–476, 2023.
- [565] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.
- [566] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 2016. PMLR.
- [567] Yarin Gal, Mark van der Wilk, and Carl E. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models, 2014.
- [568] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [569] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192. PMLR, 2017.
- [570] Zhe Gan, Changyou Chen, Ricardo Henao, David Carlson, and Lawrence Carin. Scalable deep Poisson factor analysis for topic modeling. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1823–1832, Lille, France, 2015. PMLR.
- [571] Ravi Ganti and Alexander Gray. Upal: Unbiased pool based active learning. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 422–431, La Palma, Canary Islands, 2012. PMLR.
- [572] Erdun Gao, Ignavier Ng, Mingming Gong, Li Shen, Wei Huang, Tongliang Liu, Kun Zhang, and Howard Bondell. Missdag: Causal discovery in the presence of missing data with continuous additive noise models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5024–5038. Curran Associates, Inc., 2022.
- [573] Richard Gao, Michael Deistler, and Jakob H Macke. Generalized Bayesian inference for scientific simulators via amortized cost estimation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80191–80219. Curran Associates, Inc., 2023.
- [574] Tianxiang Gao, Xiaokai Huo, Hailiang Liu, and Hongyang Gao. Infinitely deep residual networks: Unveiling wide neural ODEs as Gaussian processes, 2024.
- [575] Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Global convergence of stochastic gradient Hamiltonian Monte Carlo for nonconvex stochastic optimization: Nonasymptotic performance bounds and momentum-based acceleration. *Operations Research*, 70(5):2931–2947, 2022.

- [576] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.
- [577] Ángel F. García-Fernández, Lennart Svensson, Mark R. Morelande, and Simo Särkkä. Posterior linearization filter: Principles and implementation using sigma points. *IEEE Transactions on Signal Processing*, 63(20): 5561–5573, 2015.
- [578] Pedro J. García-Laencina, José-Luis Sancho-Gómez, and Aníbal R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.
- [579] Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. DOCKSTRING: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62(15):3486–3502, 2022.
- [580] Jacob Gardner, Geoff Pleiss, Kilian Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [581] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P. Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31, pages 8789–8798, 2018.
- [582] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 2018.
- [583] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes, 2018.
- [584] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.
- [585] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic, 2017.
- [586] Elizabeth S. Garrett and Scott L. Zeger. Latent class model diagnosis. *Biometrics*, 56(4):1055–1067, 2000.
- [587] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow Gaussian processes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [588] Gebhard, Timothy D., Wildberger, Jonas, Dax, Maximilian, Kofler, Annalena, Angerhausen, Daniel, Quanz, Sascha P., and Schölkopf, Bernhard. Flow matching for atmospheric retrieval of exoplanets: Where reliability meets adaptive noise levels. *A&A*, 693:A42, 2025.
- [589] Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11098–11116. PMLR, 2023.
- [590] Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, Agrin Hilmkil, Joel Jennings, Meyer Scetbon, Miltiadis Allamanis, and Cheng Zhang. Deep end-to-end causal inference. *Transactions on Machine Learning Research*, 2024.
- [591] Johanna Marie Gegenfurtner, Moritz Grillo, and Guido Montúfar. The symmetries of three-layer ReLU networks, 2026.
- [592] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [593] Seymour Geisser. *Predictive Inference: An Introduction*. Number 55 in Monographs on Statistics and Applied Probability. Chapman & Hall, New York, 1993.

- [594] Andrew Gelman. Parameterization and Bayesian modeling. *Journal of the American Statistical Association*, 99(466):537–545, 2004.
- [595] Andrew Gelman. Multilevel (hierarchical) modeling: What it can and cannot do. *Technometrics*, 48(3): 432–435, 2006.
- [596] Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis*, 1(3):515 – 534, 2006.
- [597] Andrew Gelman. Objections to Bayesian statistics. *Bayesian Analysis*, 3(3):445 – 449, 2008.
- [598] Andrew Gelman. David MacKay and Occam’s Razor. <https://statmodeling.stat.columbia.edu/2011/12/04/david-mackay-and-occams-razor/>, 2011. Blog post on Statistical Modeling, Causal Inference, and Social Science.
- [599] Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163 – 185, 1998.
- [600] Andrew Gelman and Donald B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457 – 472, 1992.
- [601] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.
- [602] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Akti Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Third edition edition, 2013.
- [603] Gemini Team, Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025.
- [604] Gemma Team. Gemma: Open models based on gemini research and technology, 2024.
- [605] Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 75460–75482. Curran Associates, Inc., 2025.
- [606] A.C. Genz and A.A. Malik. Remarks on algorithm 006: An adaptive algorithm for numerical integration over an n-dimensional rectangular region. *Journal of Computational and Applied Mathematics*, 6(4):295–302, 1980.
- [607] Edward I. George and Robert E. McCulloch. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.
- [608] M. Gergonne. Application de la méthode des moindres carrés à l’interpolation des suites. *Annales de Mathématiques pures et appliquées*, 1815.
- [609] Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario March, and Jean-Francis Roy. Risk bounds for the majority vote: From a PAC-Bayesian analysis to a learning algorithm. *Journal of Machine Learning Research*, 16(26):787–860, 2015.
- [610] Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [611] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Springer, 1992.
- [612] Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.
- [613] Charles J. Geyer. Introduction to Markov chain Monte Carlo. *Handbook of Markov Chain Monte Carlo*, pages 3–48, 2011.

- [614] Florin C. Ghesu, Bogdan Georgescu, Awais Mansoor, Youngjin Yoo, Eli Gibson, R.S. Vishwanath, Abishek Balachandran, James M. Balter, Yue Cao, Ramandeep Singh, Subba R. Digumorthy, Mannudeep K. Kalra, Sasa Grbic, and Dorin Comaniciu. Quantifying and leveraging predictive uncertainty for medical image assessment. *Medical Image Analysis*, 68:101855, 2021.
- [615] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pages 291–326. Chapman and Hall/CRC, 2022.
- [616] S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi. Posterior consistency of Dirichlet mixtures in density estimation. *The Annals of Statistics*, 27(1):143 – 158, 1999.
- [617] Subhashis Ghosal and Aad van der Vaart. Convergence rates of posterior distributions for noniid observations. *The Annals of Statistics*, 35(1):192 – 223, 2007.
- [618] Subhashis Ghosal and Aad Van der Vaart. *Fundamentals of nonparametric Bayesian inference*. Cambridge University Press, 2017.
- [619] Subhashis Ghosal, Jayanta K. Ghosh, and Aad W. van der Vaart. Convergence rates of posterior distributions. *The Annals of Statistics*, 28(2):500 – 531, 2000.
- [620] Avrajit Ghosh, Bai Cong, Rio Yokota, Saiprasad Ravishankar, Rongrong Wang, Molei Tao, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning finds flatter solutions at the edge of stability, 2025.
- [621] Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan S. Yedidia. Assumed density filtering methods for learning Bayesian neural networks. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1589–1595. AAAI Press, 2016.
- [622] Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1744–1753. PMLR, 2018.
- [623] Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Model selection in Bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, 20(182):1–46, 2019.
- [624] Noor Fathima Khanum Mohamed Ghouse, Jens Petersen, Auke J. Wiggers, Tianlin Xu, and Guillaume Sautiere. Neural image compression with a diffusion-based decoder, 2023.
- [625] Isaac Gibbs, John J. Cherian, and Emmanuel J. Candès. Conformal prediction with conditional guarantees, 2023.
- [626] Itzhak Gilboa and Massimo Marinacci. *Ambiguity and the Bayesian Paradigm*, pages 385–439. Springer International Publishing, Cham, 2016.
- [627] Michael B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.
- [628] Michael B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.
- [629] Walter R. Gilks and Carlo Berzuini. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.
- [630] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.
- [631] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 2002.
- [632] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

- [633] John Gkountouras, Matthias Lindemann, Phillip Lippe, Efstratios Gavves, and Ivan Titov. Language agents meet causality – bridging LLMs and causal world models, 2024.
- [634] Pierre Glaser, Michael Arbel, Samo Hromadka, Arnaud Doucet, and Arthur Gretton. Maximum likelihood learning of unnormalized models for simulation-based inference, 2022.
- [635] Manuel Glöckler, Michael Deistler, and Jakob H. Macke. Variational methods for simulation-based inference, 2022.
- [636] Manuel Gloeckler, Michael Deistler, Christian Dietrich Weillbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15735–15766. PMLR, 2024.
- [637] Manuel Gloeckler, Shoji Toyota, Kenji Fukumizu, and Jakob H. Macke. Compositional simulation-based inference for time series, 2024.
- [638] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, Volume 10 - 2019, 2019.
- [639] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.
- [640] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.
- [641] Jinwoo Go and Tobin Isaac. Robust expected information gain for optimal Bayesian experimental design using ambiguity sets. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 728–737. PMLR, 2022.
- [642] Daniel Goc and Gergely Flamich. On channel simulation with causal rejection samplers. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pages 1682–1687, 2024.
- [643] Takashi Goda, Tomohiko Hironaka, Wataru Kitade, and Adam Foster. Unbiased MLMC stochastic gradient-based optimization of Bayesian experimental designs. *SIAM Journal on Scientific Computing*, 44(1):A286–A311, 2022.
- [644] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, David S Greenberg, and Jakob H Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, 2020.
- [645] Mehmet Gönen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(64):2211–2268, 2011.
- [646] Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Sliced kernelized Stein discrepancy, 2020.
- [647] Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise, 2022.
- [648] Irving John Good. *The Estimation of Probabilities: An essay on modern Bayesian methods*. MIT Press, 1965.
- [649] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.
- [650] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [651] Leo A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61(2):215–231, 1974.

- [652] Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1292–1301. PMLR, 2017.
- [653] Jackson Gorham, Anant Raj, and Lester Mackey. Stochastic Stein discrepancies. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17931–17942. Curran Associates, Inc., 2020.
- [654] Isobel Claire Gormley and Sylvia Frühwirth-Schnatter. Mixture of experts models. In *Handbook of mixture analysis*, pages 271–307. Chapman and Hall/CRC, 2019.
- [655] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5): 9–21, 2001.
- [656] V.K. Goyal, J. Zhuang, and M. Veiterli. Transform coding with backward adaptive updates. *IEEE Transactions on Information Theory*, 46(4):1623–1633, 2000.
- [657] Robert Grande, Thomas Walsh, and Jonathan How. Sample efficient reinforcement learning with Gaussian processes. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1332–1340, Beijing, China, 2014. PMLR.
- [658] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.
- [659] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3831–3841. PMLR, 2021.
- [660] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.
- [661] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998.
- [662] Peter J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.
- [663] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2404–2414. PMLR, 2019.
- [664] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [665] Louis Grenioux, Maxence Noble, Marylou Gabri  , and Alain Oliviero Durmus. Stochastic localization via iterative posterior sampling. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16337–16376. PMLR, 2024.
- [666] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Sch  lkopf, and Alex Smola. A kernel method for the two-sample-problem. In B. Sch  lkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.
- [667] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Sch  lkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

- [668] Sam Griesemer, Defu Cao, Zijun Cui, Carolina Osorio, and Yan Liu. Active sequential posterior estimation for sample-efficient simulation-based inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 127907–127936. Curran Associates, Inc., 2024.
- [669] Nicholas Grieshop and Christopher K Wikle. Data-driven modeling of wildfire spread with stochastic cellular automata and latent spatio-temporal dynamics. *Spatial Statistics*, 59:100794, 2024.
- [670] Nicholas Grieshop and Christopher K. Wikle. Echo state network-enhanced symbolic regression for spatio-temporal binary stochastic cellular automata. *Spatial Statistics*, 60:100827, 2024.
- [671] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2):577–586, 2020.
- [672] Thomas Griffiths, Michael Jordan, Joshua Tenenbaum, and David Blei. Hierarchical topic models and the nested chinese restaurant process. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.
- [673] Christopher Heje Grønbech, Maximillian Fornitz Vording, Pascal N Timshel, Casper Kaae Sønderby, Tune H Pers, and Ole Winther. scvae: variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, 2020.
- [674] Lukas N. Groschner, Jonatan G. Malis, Birte Zuidinga, and Alexander Borst. A biophysical account of multiplication by a single neuron. *Nature*, 603(7899):119–123, 2022.
- [675] Richard Grumitt, Biwei Dai, and Uros Seljak. Deterministic Langevin Monte Carlo with normalizing flows for Bayesian inference. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11629–11641. Curran Associates, Inc., 2022.
- [676] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.
- [677] Peter D. Grünwald and A. Philip Dawid. Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. *The Annals of Statistics*, 32(4):1367 – 1433, 2004.
- [678] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487. Curran Associates, Inc., 2020.
- [679] Yuqi Gu and David B Dunson. Bayesian pyramids: identifiable multilayer discrete latent structure models for discrete data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(2):399–426, 2023.
- [680] Yuqi Gu, Elena E. Erosheva, Gongjun Xu, and David B. Dunson. Dimension-grouped mixed membership models for multivariate categorical data. *Journal of Machine Learning Research*, 24(88):1–49, 2023.
- [681] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 265–272. ACM, 2005.
- [682] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [683] Arthur Guez, David Silver, and Peter Dayan. Sample-based search for Bayes-adaptive reinforcement learning. *Journal of Artificial Intelligence Research*, 48:841–883, 2013.
- [684] Pierre-Antoine Guihéneuf. Dynamical properties of spatial discretizations of a generic homeomorphism. *Ergodic Theory and Dynamical Systems*, 35(5):1474–1523, 2015.
- [685] Cole Christian Gulino and Alexander Rashid Ansari. Providing actionable uncertainties in autonomous vehicles, 2024. US Patent 11,860,636.

- [686] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [687] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.
- [688] Zongyu Guo, Gergely Flamich, Jiajun He, Zhibo Chen, and José Miguel Hernández-Lobato. Compression with Bayesian implicit neural representations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 1938–1956. Curran Associates, Inc., 2023.
- [689] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1737–1746, Lille, France, 2015. PMLR.
- [690] Tarun Gupta, Wenbo Gong, Chao Ma, Nick Pawlowski, Agrin Hilmkil, Meyer Scetbon, Marc Rigter, Ade Famoti, Ashley Juan Llorens, Jianfeng Gao, Stefan Bauer, Danica Kragic, Bernhard Schölkopf, and Cheng Zhang. The essential role of causality in foundation world models for embodied ai, 2024.
- [691] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1842–1850. PMLR, 2018.
- [692] Fredrik K. Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 318–319, 2020.
- [693] Mats Gyllenberg, Timo Koski, Edwin Reilink, and Martin Verlaan. Non-uniqueness in probabilistic numerical identification of bacteria. *Journal of Applied Probability*, 31(2):542–548, 1994.
- [694] Vu A. Ha and Peter Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In Dan Geiger and Prakash P. Shenoy, editors, *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pages 215–222. Morgan Kaufmann, 1997.
- [695] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S. Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M. Brockway, and Alán Aspuru-Guzik. The harvard clean energy project: Large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, 2011.
- [696] Ian Hacking. *Experimentation and scientific realism*. 1982.
- [697] Ian Hacking. *Representing and Intervening: Introductory Topics in the Philosophy of Natural Science*. Cambridge University Press, 1983.
- [698] Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8175–8195. PMLR, 2022.
- [699] Maxime Haddouche, Paul Viallard, Umut Simsekli, and Benjamin Guedj. A PAC-Bayesian link between generalisation and flat minima. In Gautam Kamath and Po-Ling Loh, editors, *Proceedings of The 36th International Conference on Algorithmic Learning Theory*, volume 272 of *Proceedings of Machine Learning Research*, pages 481–511. PMLR, 2025.
- [700] ChangHoon Hahn and Peter Melchior. Accelerated Bayesian sed modeling using amortized neural posterior estimation. *The Astrophysical Journal*, 938(1):11, 2022.

- [701] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.
- [702] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2009.
- [703] Marshall Hall. *Combinatorial theory*. John Wiley & Sons, second edition edition, 2011.
- [704] Xing Han, Huy Nguyen, Carl Harris, Nhat Ho, and Suchi Saria. Fusemoe: Mixture-of-experts transformers for fleximodal fusion. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 67850–67900. Curran Associates, Inc., 2024.
- [705] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
- [706] Stijn Hantson. California daily fire growth database (2012-2021), 2020.
- [707] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. Asgn: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’20, pages 731–752, New York, NY, USA, 2020. Association for Computing Machinery.
- [708] Rolland L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research (1896-1977)*, 76(8):1905–1915, 1971.
- [709] James Harrison, John Willes, and Jasper Snoek. Variational Bayesian last layers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 54251–54280, 2024.
- [710] James Harrison, John Willes, Paul Brunzema, and Jasper Snoek. Heteroscedastic variational last layers. In *7th Symposium on Advances in Approximate Bayesian Inference— Workshop Track*, 2025.
- [711] Allen G. Hart, James L. Hook, and Jonathan H.P. Dawes. Echo state networks trained by Tikhonov least squares are $l^2(\mu)$ approximators of ergodic dynamical systems. *Physica D: Nonlinear Phenomena*, 421: 132882, 2021.
- [712] Meysam Hashemi, Anirudh N. Vattikonda, Jayant Jha, Viktor Sip, Marmaduke M. Woodman, Fabrice Bartolomei, and Viktor K. Jirsa. Amortized Bayesian inference on generative dynamical network models of epilepsy using deep neural density estimators. *Neural Networks*, 163:178–194, 2023.
- [713] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [714] Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [715] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal random code learning: Getting bits back from compressed model parameters. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [716] Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Variable length discrete flow matching with sequence-level edit operations. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 139037–139070. Curran Associates, Inc., 2025.
- [717] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. DSelect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29335–29347. Curran Associates, Inc., 2021.

- [718] Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1010–1022. Curran Associates, Inc., 2020.
- [719] Jiajun He, Gergely Flamich, Zongyu Guo, and José Miguel Hernández Lobato. Recombiner: Robust and enhanced compression with Bayesian implicit neural representations. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 33185–33211, 2024.
- [720] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.
- [721] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [722] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.
- [723] Zhiwei He, Xing Wang, Wenxiang Jiao, Zhuosheng Zhang, Rui Wang, Shuming Shi, and Zhaopeng Tu. Improving machine translation with human feedback: An exploration of quality estimation as a reward model. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8164–8180, Mexico City, Mexico, 2024. Association for Computational Linguistics.
- [724] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In Rolf ECKMILLER, editor, *Advanced Neural Computers*, pages 129–135. North-Holland, Amsterdam, 1990.
- [725] Marcel Hedman, Desi R. Ivanova, Cong Guan, and Tom Rainforth. Step-DAD: Semi-amortized policy-based Bayesian experimental design. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 22904–22923. PMLR, 2025.
- [726] Marcel Hedman, Emily Alger, Brieuc Lehmann, Chris Holmes, and Tom Rainforth. Fairbed: A Bayesian experimental design approach to gathering fairer data, 2026.
- [727] Mahmoud Hegazy and Cheuk Ting Li. Randomized quantization with exact error distribution. In *2022 IEEE Information Theory Workshop (ITW)*, pages 350–355. IEEE, 2022.
- [728] Pashupati Hegde, Markus Heinonen, Harri Lähdesmäki, and Samuel Kaski. Deep learning with differential Gaussian process flows. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1812–1821. PMLR, 2019.
- [729] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [730] Stefan Heinrich. Multilevel Monte Carlo methods. In *Large-Scale Scientific Computing: Third International Conference, LSSC 2001 Sozopol, Bulgaria, June 6–10, 2001 Revised Papers 3*, pages 58–67. Springer, 2001.
- [731] Kai Helli, David Schnurr, Noah Hollmann, Samuel Müller, and Frank Hutter. Drift-resilient tabpfn: In-context learning temporal distribution shifts on tabular data. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 98742–98781. Curran Associates, Inc., 2024.
- [732] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: a survey. *Machine Learning*, 113(5):3073–3110, 2024.

- [733] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations, 2019.
- [734] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- [735] James Hensman and Neil D. Lawrence. Nested variational compression in deep Gaussian processes, 2014.
- [736] James Hensman, Magnus Rattray, and Neil Lawrence. Fast variational inference in the conjugate exponential family. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [737] James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI’13, pages 282–290, Arlington, Virginia, USA, 2013. AUAI Press.
- [738] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA, 2015. PMLR.
- [739] James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [740] James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018.
- [741] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4239–4248. PMLR, 2020.
- [742] Joeri Hermans, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, Volodimir Begy, and Gilles Louppe. A crisis in simulation-based inference? beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*, 2022.
- [743] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France, 2015. PMLR.
- [744] Gonzalo Hernández-Muñoz, Carlos Villacampa-Calvo, and Daniel Hernández-Lobato. Deep Gaussian processes using expectation propagation and Monte Carlo methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 479–494. Springer, 2020.
- [745] Edwin Hewitt and Leonard J Savage. Symmetric measures on cartesian products. *Transactions of the American Mathematical Society*, 80(2):470–501, 1955.
- [746] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [747] Spencer Hill, Fady Alajaji, Tamás Linder, and Gergely Flamich. Rejection sampling is optimal for relative entropy coding, 2026. To appear at the 2026 IEEE Information Theory Workshop (ITW).
- [748] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

- [749] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.
- [750] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- [751] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.
- [752] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [753] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.
- [754] Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [755] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.
- [756] Nhat Ho, Chiao-Yu Yang, and Michael I. Jordan. Convergence rates for Gaussian mixtures of experts. *Journal of Machine Learning Research*, 23(323):1–81, 2022.
- [757] Viet Ha Hoang, Christoph Schwab, and Andrew M Stuart. Complexity analysis of accelerated MCMC methods for Bayesian inversion. *Inverse Problems*, 29(8):085010, 2013.
- [758] Vinh Hoang, Luis Espath, Sebastian Krumscheid, and Raúl Tempone. Scalable method for Bayesian experimental design without integrating over posterior distribution. *SIAM/ASA Journal on Uncertainty Quantification*, 13(1):114–139, 2025.
- [759] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.
- [760] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [761] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- [762] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47):1593–1623, 2014.
- [763] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(40):1303–1347, 2013.
- [764] Matthew Douglas Hoffman. Roundoff error in metropolis-hastings accept-reject steps. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.
- [765] Clara Hoffmann and Nadja Klein. Marginally calibrated response distributions for end-to-end learning in autonomous driving. *The Annals of Applied Statistics*, 17(2):1740–1763, 2023.
- [766] Paul Hofman, Yusuf Sale, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty: A credal approach. In *ICML 2024 Workshop on Structured Probabilistic Inference* $\{\backslash\mathcal{E}\}$ *Generative Modeling*, 2024.
- [767] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web, WWW '06*, pages 633–642, New York, NY, USA, 2006. Association for Computing Machinery.
- [768] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

- [769] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [770] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Scalable, generalizable, and accurate tabular predictions with TabPFN v2. *Nature*, 2025. In press.
- [771] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeyer, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.
- [772] C. C. Holmes and S. G. Walker. Assigning a value to a power likelihood in a general Bayesian model. *Biometrika*, 104(2):497–503, 2017.
- [773] Chris C. Holmes and Stephen G. Walker. Statistical inference with exchangeability and martingales. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2247):20220143, 2023.
- [774] Philip Holmes, John L. Lumley, and Gal Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs on Mechanics. Cambridge University Press, second edition edition, 2012.
- [775] Shenda Hong, Yanbo Xu, Alind Khare, Satria Priambada, Kevin Maher, Alaa Aljiffry, Jimeng Sun, and Alexey Tumanov. Holmes: health online model ensemble serving for deep learning models in intensive care units. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1614–1624, 2020.
- [776] A. Honkela and H. Valpola. Variational learning and bits-back coding: an information-theoretic view to Bayesian learning. *IEEE Transactions on Neural Networks*, 15(4):800–810, 2004.
- [777] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending tabpfn-v2 to time series forecasting, 2025.
- [778] Emiel Hoogetboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [779] Emiel Hoogetboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models, 2021.
- [780] Emiel Hoogetboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12454–12465. Curran Associates, Inc., 2021.
- [781] Emiel Hoogetboom, Eirikur Agustsson, Fabian Mentzer, Luca Versari, George Toderici, and Lucas Theis. High-fidelity image compression with score-based generative models, 2023.
- [782] Jesse Hoogland and Stan van Wingerden. You're measuring model complexity wrong, 2023.
- [783] Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy and stagewise development in transformers. *Transactions on Machine Learning Research*, 2025.
- [784] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011.
- [785] Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Jose Hernández-lobato. Collaborative Gaussian processes for preference learning. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [786] Patrik Hoyer, Dominik Janzing, Joris Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

- [787] Eirik Høyheim, Lars Skaaret-Lund, Solve Sæbø, and Aliaksandr Hubin. Explainable Bayesian deep learning through input-skip latent binary Bayesian neural networks. *The journal of artificial intelligence research*, 86:1–42, 2026.
- [788] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021.
- [789] Xun Huan and Youssef M. Marzouk. Sequential Bayesian optimal experimental design via approximate dynamic programming, 2016.
- [790] Xun Huan, Jayanth Jagalur, and Youssef Marzouk. Optimal experimental design: Formulations and computations. *Acta Numerica*, 33:715–840, 2024.
- [791] Daolang Huang, Ayush Bharti, Amauri Souza, Luigi Acerbi, and Samuel Kaski. Learning robust statistics for simulation-based inference under model misspecification. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 7289–7310. Curran Associates, Inc., 2023.
- [792] Daolang Huang, Yujia Guo, Luigi Acerbi, and Samuel Kaski. Amortized Bayesian experimental design for decision-making. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 109460–109486. Curran Associates, Inc., 2024.
- [793] Daolang Huang, Xinyi Wen, Ayush Bharti, Samuel Kaski, and Luigi Acerbi. Aline: Joint amortization for Bayesian inference and active data acquisition. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 54068–54102. Curran Associates, Inc., 2025.
- [794] Daolang Huang, Zhuoyue Huang, Conor Hassan, Luigi Acerbi, Samuel Kaski, and Tom Rainforth. Efficient adaptive data acquisition via pretrained belief representations, 2026.
- [795] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [796] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4542–4551. PMLR, 2020.
- [797] Lei Huang, Meng Song, Hui Shen, Huixiao Hong, Ping Gong, Hong-Wen Deng, and Chaoyang Zhang. Deep learning methods for omics data imputation. *Biology*, 12(10), 2023.
- [798] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks, 2020.
- [799] Zhuoyue Huang, Freddie Bickford Smith, and Tom Rainforth. Loss-driven Bayesian active learning. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.
- [800] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [801] Aliaksandr Hubin and Geir Storvik. Combining model and parameter uncertainty in Bayesian neural networks, 2019.
- [802] Aliaksandr Hubin and Geir Storvik. Sparse Bayesian neural networks: bridging model and parameter uncertainty through scalable variational inference. *Mathematics*, 12(6):788, 2024.
- [803] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
- [804] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

- [805] Raphaël Huser and Jennifer L Wadsworth. Advances in statistical modeling of spatial extremes. *Wiley Interdisciplinary Reviews: Computational Statistics*, 14(1):e1537, 2022.
- [806] Rob J. Hyndman. Computing and graphing highest density regions. *The American Statistician*, 50(2):120–126, 1996.
- [807] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.
- [808] Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O. Hoyer. Estimation of a structural vector autoregression model using non-gaussianity. *Journal of Machine Learning Research*, 11(56):1709–1731, 2010.
- [809] Shun ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, 1993.
- [810] Christian Igel and Michael Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.
- [811] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [812] Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Khan Mohammad Emtiyaz. Scalable marginal likelihood estimation for model selection in deep learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4563–4573. PMLR, 2021.
- [813] Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of Bayesian neural nets via local linearization. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 703–711. PMLR, 2021.
- [814] Alexander Immer, Lucas Torroba Hennigen, Vincent Fortuin, and Ryan Cotterell. Probing as quantifying inductive bias. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 60, 2022.
- [815] Alexander Immer, Tycho van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable Laplace approximations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 12449–12463. Curran Associates, Inc., 2022.
- [816] Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 3(25):602, 2018.
- [817] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.
- [818] Jacopo Iollo, Christophe Heinkelé, Pierre Alliez, and Florence Forbes. PASOA- PArticle baSed Bayesian optimal adaptive design. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21020–21046. PMLR, 2024.
- [819] Jacopo Iollo, Christophe Heinkelé, Pierre Alliez, and Florence Forbes. Bayesian experimental design via contrastive diffusions. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 73125–73151, 2025.
- [820] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. not-miwa: Deep generative modelling with missing not at random data, 2020.

- [821] Sahel Iqbal, Adrien Corenflos, Simo Särkkä, and Hany Abdulsamad. Nesting particle filters for experimental design in dynamical systems. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21047–21068. PMLR, 2024.
- [822] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoeffler. Data movement is all you need: A case study on optimizing transformers. *Proceedings of Machine Learning and Systems*, 3:711–732, 2021.
- [823] Desi R Ivanova, Adam Foster, Steven Kleinegesse, Michael Gutmann, and Thomas Rainforth. Implicit deep adaptive design: Policy-based experimental design without likelihoods. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25785–25798. Curran Associates, Inc., 2021.
- [824] Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press, 2018.
- [825] Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 1169–1179. PMLR, 2020.
- [826] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are Bayesian neural network posteriors really like? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4629–4640. PMLR, 2021.
- [827] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.
- [828] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
- [829] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [830] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R*. Springer, second edition edition, 2021.
- [831] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016.
- [832] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [833] Sanket Jantre, Shrijita Bhattacharya, and Tapabrata Maiti. Spike-and-slab shrinkage priors for structurally sparse Bayesian neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6): 11176–11188, 2025.
- [834] David Janz, Jos van der Westhuizen, and José Miguel Hernández-Lobato. Actively learning what makes a discrete sequence valid. *Workshop on Principled Approaches to Deep Learning, International Conference on Machine Learning*, 2017.
- [835] David Janz, Jos van der Westhuizen, Brooks Paige, Matt J. Kusner, and José Miguel Hernández-Lobato. Learning a generative model for validity in complex discrete structures. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

- [836] David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschitschek. Successor uncertainties: Exploration and uncertainty in temporal difference learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [837] C. Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Phys. Rev. E*, 56:5018–5035, 1997.
- [838] Neal Jean, Sang Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [839] Alan Jeffrey. *Linear algebra and ordinary differential equations*. Blackwell Scientific, 1990.
- [840] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.
- [841] Abebech Jenber Belay, Yelkal Muluaem Walle, and Melaku Bitew Haile. Deep ensemble learning and quantum machine learning approach for alzheimer’s disease detection. *Scientific Reports*, 14(1):14196, 2024.
- [842] Jeon. ThompsonBALD: a new approach to Bayesian batch active learning for deep learning via Thompson sampling. Master’s thesis, University College London, 2020.
- [843] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.
- [844] J Jiang. Image compression with neural networks—a survey. *Signal processing: image Communication*, 14(9):737–760, 1999.
- [845] Jiashun Jin, Zheng Tracy Ke, and Shengming Luo. Mixed membership estimation for social networks. *Journal of Econometrics*, 2023.
- [846] Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng LYU, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. Cladder: Assessing causal reasoning in language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 31038–31065. Curran Associates, Inc., 2023.
- [847] Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression, 2019.
- [848] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.
- [849] Matt Jones, Peter Chang, and Kevin Murphy. Bayesian online natural gradient (bong). In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 131104–131153. Curran Associates, Inc., 2024.
- [850] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.
- [851] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
- [852] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

- [853] Maxwell B Joseph, Matthew W Rossi, Nathan P Mietkiewicz, Adam L Mahood, Megan E Cattau, Lise Ann St. Denis, R Chelsea Nagy, Virginia Iglesias, John T Abatzoglou, and Jennifer K Balch. Spatiotemporal prediction of wildfire size extremes with Bayesian finite sample maxima. *Ecological Applications*, 29(6): e01898, 2019.
- [854] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on Bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.
- [855] AG Journel and Ch J Huijbregts. *Mining Geostatistics*. Academic Press, 1978.
- [856] Ngumbang Juat, Mike Meredith, and John Kruschke. Package ‘hdinterval’, 2022. Accessed on May 9, 2023.
- [857] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference - ACC’95*, volume 3, pages 1628–1632 vol.3, 1995.
- [858] Joseph Kadane and Lara J. Wolfson. Experiences in elicitation. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1):3–19, 1998.
- [859] Peter Kairouz and H. Brendan McMahan. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 2021.
- [860] Henry F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3): 187–200, 1958.
- [861] Olav Kallenberg. Spreading and Predictable Sampling in Exchangeable Sequences and Processes. *The Annals of Probability*, 16(2):508 – 534, 1988.
- [862] Olav Kallenberg. *Foundations of Modern Probability*. Springer, 1997.
- [863] Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1701–1710. PMLR, 2018.
- [864] Sanket Kamthe, So Takao, Shakir Mohamed, and Marc Peter Deisenroth. Iterative state estimation in non-linear dynamical systems using approximate expectation propagation. *Transactions on Machine Learning Research*, 2022.
- [865] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K. Sriperumbudur. Gaussian processes and reproducing kernels: Connections and equivalences, 2025.
- [866] Genki N Kanda, Taku Tsuzuki, Motoki Terada, Noriko Sakai, Naohiro Motozawa, Tomohiro Masuda, Mitsuhiro Nishida, Chihaya T Watanabe, Tatsuki Higashi, Shuhei A Horiguchi, Taku Kudo, Motohisa Kamei, Genshiro A Sunagawa, Kenji Matsukuma, Takeshi Sakurada, Yosuke Ozawa, Masayo Takahashi, Koichi Takahashi, and Tohru Natsume. Robotic search for optimal cell culture in regenerative medicine. *eLife*, 11:e77007, 2022.
- [867] Dae Y Kang, Pamela N DeYoung, Justin Tantiengloc, Todd P Coleman, and Robert L Owens. Statistical uncertainty quantification to augment clinical decision support: a first implementation in sleep medicine. *NPJ digital medicine*, 4(1):142, 2021.
- [868] Leonid Vitaliyevich Kantorovich. On the transfer of masses. *Doklady Akademii Nauk SSSR*, 37:227–229, 1942.
- [869] Leonid Vitaliyevich Kantorovich. On a problem of Monge. *Uspekhi Matematicheskikh Nauk*, 3(2):225–226, 1948.
- [870] Sanyam Kapoor, Wesley J. Maddox, Pavel Izmailov, and Andrew Gordon Wilson. On uncertainty, tempering, and data augmentation in Bayesian classification, 2022.
- [871] Mohammad Reza Karimi Jaghargh, Ya-Ping Hsieh, and Andreas Krause. Stochastic approximation algorithms for systems of interacting particles. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 55826–55847. Curran Associates, Inc., 2023.

- [872] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [873] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577. Curran Associates, Inc., 2022.
- [874] Mikolaj J. Kasprzak, Ryan Giordano, and Tamara Broderick. How good is your Laplace approximation of the Bayesian posterior? finite-sample computable error bounds for a variety of useful divergences. *Journal of Machine Learning Research*, 26(87):1–81, 2025.
- [875] Anya Katsevich and Philippe Rigollet. On the approximation accuracy of Gaussian variational inference. *The Annals of Statistics*, 52(4):1384 – 1409, 2024.
- [876] Ramneet Kaur, Xiayan Ji, Souradeep Dutta, Michele Caprio, Yahan Yang, Elena Bernardis, Oleg Sokolsky, and Insup Lee. Using semantic information for defining and detecting ood inputs, 2023.
- [877] John A Keith, Valentin Vassilev-Galindo, Bingqing Cheng, Stefan Chmiela, Michael Gastegger, Klaus-Robert Muller, and Alexandre Tkatchenko. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chemical reviews*, 121(16):9816–9872, 2021.
- [878] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository, 2025.
- [879] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [880] MC Kennedy and A O’Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.
- [881] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 2018.
- [882] Mohammad Emtiyaz Khan. Knowledge adaptation as posterior correction, 2025.
- [883] Mohammad Emtiyaz Khan and Håvard Rue. The Bayesian learning rule. *Journal of Machine Learning Research*, 24(281):1–46, 2023.
- [884] Mohammad Emtiyaz Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into Gaussian processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [885] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2023.
- [886] J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society: Series B (Methodological)*, 21(2):272–304, 1959.
- [887] Jaeyeon Kim, Lee Kit, Carles Domingo i Enrich, Yilun Du, Sham Kakade, Timothy Ngatiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 40151–40183, 2026.
- [888] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5275–5285. PMLR, 2020.

- [889] Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Yakun Sophia Shao, and Amir Gholami. Full stack optimization of transformer inference: a survey, 2023.
- [890] Taehwan Kim and Brian R. King. Time series prediction using deep echo state networks. *Neural Computing and Applications*, 32(23):17769–17787, 2020.
- [891] George S. Kimeldorf and Grace Wahba. A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines. *The Annals of Mathematical Statistics*, 41(2):495 – 502, 1970.
- [892] Kingma and Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.
- [893] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021.
- [894] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.
- [895] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders, 2019.
- [896] Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3408–3417. PMLR, 2019.
- [897] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- [898] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep Bayesian active learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [899] Andreas Kirsch, Tom Rainforth, and Yarin Gal. Test distribution-aware active learning: A principled approach against distribution shift and outliers, 2021.
- [900] Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frédéric Branchaud-Charron, and Yarin Gal. Stochastic batch acquisition: A simple baseline for deep active learning. *Transactions on Machine Learning Research*, 2023. Expert Certification.
- [901] Nadja Klein and Thomas Kneib. Scale-Dependent Priors for Variance Parameters in Structured Additive Distributional Regression. *Bayesian Analysis*, 11(4):1071 – 1106, 2016.
- [902] Nadja Klein, David J Nott, and Michael Stanley Smith. Marginally calibrated deep distributional regression. *Journal of Computational and Graphical Statistics*, 30(2):467–483, 2021.
- [903] Steven Kleinegesse and Michael U. Gutmann. Bayesian experimental design for implicit models by mutual information neural estimation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5316–5326. PMLR, 2020.
- [904] Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. An optimization-centric view on Bayes’ rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 23(132):1–109, 2022.
- [905] Rik Knowles and Tom Rainforth. Tractable expected information gains for exponential family posteriors. *International Conference on Machine Learning*, 2026.
- [906] Donald Knuth and Andrew Yao. The complexity of nonuniform random number generation. *Algorithms and Complexity, New Directions and Results*, 1976.

- [907] Andrew V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.
- [908] Julius Kobińska, Emanuel Sommer, Chris Kolb, Juntae Kwon, Daniel Dold, and David Rügamer. On the Interplay of Priors and Overparametrization in Bayesian Neural Network Posteriors. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research. PMLR, 2026.
- [909] Szymon Kobus, Lucas Theis, and Deniz Gündüz. Gaussian channel simulation with rotated dithered quantization. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pages 1907–1912. IEEE, 2024.
- [910] Jonathan Koh, François Pimont, Jean-Luc Dupuy, and Thomas Opitz. Spatiotemporal wildfire modeling through point processes with moderate and extreme marks. *The annals of applied statistics*, 17(1):560–582, 2023.
- [911] Michael Kohler and Sophie Langer. On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, 49(4):2231 – 2249, 2021.
- [912] Andrey Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, 1933.
- [913] J. Zico Kolter and Andrew Y. Ng. Near-Bayesian exploration in polynomial time. In Andrea Pohoreckýj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 513–520. ACM, 2009.
- [914] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2747–2755. PMLR, 2018.
- [915] Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.
- [916] Insung Kong and Yongdai Kim. Posterior concentrations of fully-connected Bayesian neural networks with general priors on the weights, 2024.
- [917] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the metropolis-hastings budget. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 181–189, Beijing, China, 2014. PMLR.
- [918] Anna Korba, Adil Salim, Michael Arbel, Giulia Luise, and Arthur Gretton. A non-asymptotic analysis for Stein variational gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4672–4682. Curran Associates, Inc., 2020.
- [919] Anna Korba, Pierre-Cyril Aubin-Frankowski, Szymon Majewski, and Pierre Ablin. Kernel Stein discrepancy descent. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5719–5730. PMLR, 2021.
- [920] Kevin Korfmann, Oscar E Gaggiotti, and Matteo Fumagalli. *Deep learning in population genetics*, volume 15, page evad008. Oxford University Press US, 2023.
- [921] Jannik Kossen. *Advancing data-efficient deep learning: non-parametric transformers, active testing, and in-context learning*. PhD thesis, University of Oxford, 2024.
- [922] Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5753–5763. PMLR, 2021.

- [923] Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Thomas Rainforth. Active surrogate estimators: An active learning approach to label-efficient model evaluation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24557–24570. Curran Associates, Inc., 2022.
- [924] Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.
- [925] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(8):235–284, 2008.
- [926] Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18237–18248. Curran Associates, Inc., 2020.
- [927] Ranganath Krishnan, Mahesh Subedar, and Omesh Tickoo. Specifying weight priors in Bayesian deep neural networks with empirical Bayes. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4477–4484. AAAI Press, 2020.
- [928] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, even just a bit, fixes overconfidence in ReLU networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5436–5446. PMLR, 2020.
- [929] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. An infinite-feature extension for Bayesian ReLU nets that fixes their asymptotic overconfidence. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18789–18800. Curran Associates, Inc., 2021.
- [930] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Learnable uncertainty under Laplace approximations. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 2021.
- [931] Agustinus Kristiadi, Felix Dangel, and Philipp Hennig. The geometry of neural nets' parameter spaces under reparametrization. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 17669–17688. Curran Associates, Inc., 2023.
- [932] Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, and Vincent Fortuin. Promises and pitfalls of the linearized Laplace in Bayesian optimization, 2023.
- [933] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.
- [934] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). Available at <http://www.cs.toronto.edu/~kriz/cifar.html>, 2009.
- [935] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [936] Anders Krogh and Peter Sollich. Statistical mechanics of ensemble learning. *Phys. Rev. E*, 55:811–825, 1997.
- [937] Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and Its Applications*, 18(2):95–138, 1977.

- [938] Arun Kumar Kuchibhotla and Abhishek Chakraborty. Moving beyond sub-gaussianity in high-dimensional statistics: Applications in covariance estimation and linear regression. *Information and Inference: A Journal of the IMA*, 11(4):1389–1456, 2022.
- [939] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.
- [940] Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference, 2021.
- [941] Lars Kühnischel, Jerry M. Huang, Valentin Pratz, Jonas Arruda, Hans Olischläger, Daniel Habermann, Simon Kucharsky, Lasse Elsemüller, Aayush Mishra, Niels Bracher, Svenja Jedhoff, Marvin Schmitt, Paul-Christian Bürkner, and Stefan T. Radev. Bayesflow 2: Multi-backend amortized Bayesian inference in python, 2026.
- [942] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [943] Alexander Kukush. *Gaussian measures in Hilbert space: construction and properties*. John Wiley & Sons, 2020.
- [944] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.
- [945] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.
- [946] Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. Arviz a unified library for exploratory analysis of Bayesian models in python. *Journal of Open Source Software*, 4(33):1143, 2019.
- [947] Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176, Boston, Massachusetts, USA, 2004. Association for Computational Linguistics.
- [948] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [949] Julius Kunze, Daniel Severo, Jan-Willem van de Meent, and James Townsend. Practical shuffle coding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 84081–84113. Curran Associates, Inc., 2024.
- [950] Julius Kunze, Daniel Severo, giulio zani, Jan-Willem van de Meent, and James Townsend. Entropy coding of unordered data structures. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 145–161, 2024.
- [951] Richard Kurle, Botond Cseke, Alexej Klushyn, Patrick van der Smagt, and Stephan Günnemann. Continual learning with Bayesian neural networks for non-stationary data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [952] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [953] Věra Kůrková and Paul C. Kainen. Functionally equivalent feedforward neural networks. *Neural Computation*, 6(3):543–558, 1994.
- [954] Alexandre Lacasse, François Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

- [955] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning, 2019.
- [956] Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated Bayesian. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 416–424, Fort Lauderdale, FL, USA, 2011. PMLR.
- [957] Adrien Lafage, Mathieu Barbier, Gianni Franchi, and David Filliat. Hierarchical light transformer ensembles for multimodal trajectory forecasting. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pages 1682–1691, 2025.
- [958] Adrien Lafage, Gianni Franchi, Mathieu Barbier, and David Filliat. Robust trajectory forecasting in autonomous systems using mixtures of Student’s t-distributions with t-distnet. *Pattern Recognition*, 165: 111524, 2025.
- [959] Adrien Lafage, Olivier Laurent, Firas Gabetni, and Gianni Franchi. Torch-uncertainty: Deep learning uncertainty quantification. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.
- [960] John Lafferty and David Blei. Correlated topic models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.
- [961] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.
- [962] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [963] Vidhi Lalchand, Wessel Bruinsma, David Burt, and Carl Edward Rasmussen. Sparse Gaussian process hyperparameters: Optimize or integrate? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16612–16623. Curran Associates, Inc., 2022.
- [964] Ben Lambert and Aki Vehtari. R^* : A Robust MCMC Convergence Diagnostic with Uncertainty Using Decision Tree Classifiers. *Bayesian Analysis*, 17(2):353 – 379, 2022.
- [965] Marc Lambert, Silvére Bonnabel, and Francis Bach. The recursive variational Gaussian approximation (r-vga). *Statistics and Computing*, 32(1):10, 2021.
- [966] Marc Lambert, Silvére Bonnabel, and Francis Bach. The limited-memory recursive variational Gaussian approximation (L-RVGA). *Statistics and Computing*, 33(70), 2023.
- [967] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxu Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.
- [968] Jouko Lampinen and Aki Vehtari. Bayesian approach for neural networks—review and case studies. *Neural Networks*, 14(3):257–274, 2001.
- [969] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45(4):255–282, 1950.
- [970] David A Lane and William D Sudderth. Coherent predictive inference. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 166–185, 1984.
- [971] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

- [972] Kevin J. Lang and Eric B. Baum. Query learning can work poorly when a human oracle is used. *International Joint Conference on Neural Networks*, 1992.
- [973] John Langford and John Shawe-Taylor. PAC-Bayes & margins. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.
- [974] Pierre Simon Laplace. *Théorie analytique des probabilités*. Courcier, 1820.
- [975] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 29–37, Fort Lauderdale, FL, USA, 2011. PMLR.
- [976] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1558–1566, New York, New York, USA, 2016. PMLR.
- [977] J. Larsen, L.K. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71, 1996.
- [978] Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and Susan Wei. The local learning coefficient: A singularity-aware complexity measure. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 244–252. PMLR, 2025.
- [979] Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean-Marc Martinez, Andrei Bursuc, and Gianni Franchi. Packed ensembles for efficient uncertainty estimation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [980] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of Bayesian neural network posteriors. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 19067–19114, 2024.
- [981] Max-Heinrich Laves, Sontje Ihler, Jacob F. Fast, Lüder A. Kahrs, and Tobias Ortmaier. Well-calibrated regression uncertainty in medical imaging with deep learning. In Tal Arbel, Ismail Ben Ayed, Marleen de Bruijne, Maxime Descoteaux, Herve Lombaert, and Christopher Pal, editors, *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, volume 121 of *Proceedings of Machine Learning Research*, pages 393–412. PMLR, 2020.
- [982] François Laviolette, Emilie Morvant, Liva Ralaivola, and Jean-Francis Roy. Risk upper bounds for general ensemble methods with an application to multiclass classification. *Neurocomputing*, 219:15–25, 2017.
- [983] Elizabeth S Lawler and Benjamin A Shaby. Anthropogenic and meteorological effects on the counts and sizes of moderate and extreme wildfires. *Environmetrics*, 35(7):e2873, 2024.
- [984] Neil D. Lawrence. *Variational Inference in Probabilistic Models*. Ph.D. thesis, Department of Computer Science and Technology, University of Cambridge, Cambridge, UK, 2000.
- [985] Neil D. Lawrence and Andrew J. Moore. Hierarchical Gaussian process latent variable models. In *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, pages 481–488, New York, NY, USA, 2007. Association for Computing Machinery.
- [986] Miguel Lázaro-Gredilla. Bayesian warped Gaussian processes. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [987] Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

- [988] Miguel Lazaro-Gredilla and Aníbal R. Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE Transactions on Neural Networks*, 21(8):1345–1351, 2010.
- [989] Minh Le, An Nguyen, Huy Nguyen, Trang Nguyen, Trang Pham, Linh Van Ngo, and Nhat Ho. Mixture of experts meets prompt-based continual learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 119025–119062. Curran Associates, Inc., 2024.
- [990] Minh Le, Chau Nguyen, Huy Nguyen, Quyen Tran, Trung Le, and Nhat Ho. Revisiting prefix-tuning: Statistical benefits of reparameterization among prompts. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 47534–47575, 2025.
- [991] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference Compilation and Universal Probabilistic Programming. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348. PMLR, 2017.
- [992] Judith Lean. Solar irradiance reconstruction. In *Data contribution series 2004-035, IGBP PAGES/World Data Center for Paleoclimatology NOAA/NGDC Paleoclimatology Program, Boulder, CO, USA*, 2004.
- [993] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [994] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.
- [995] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2010.
- [996] Emanuele Ledda, Giorgio Fumera, and Fabio Roli. Dropout injection at test time for post hoc uncertainty quantification in neural networks. *Information Sciences*, 645:119356, 2023.
- [997] Anthony Lee, Christopher Yau, Michael B Giles, Arnaud Doucet, and Christopher C Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of computational and graphical statistics*, 19(4):769–789, 2010.
- [998] Donggyu Lee, Hyeok Yun, Meeyoung Cha, Sungwon Park, Sangyoon Park, and Jihee Kim. Econcausal: A context-aware economic reasoning benchmark for large language models, 2025.
- [999] Herbert KH Lee. Neural networks and default priors. In *Proceedings of the American Statistical Association, Section on Bayesian Statistical Science*, volume 4, 2005.
- [1000] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes, 2017.
- [1001] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15156–15172. Curran Associates, Inc., 2020.
- [1002] Jongseok Lee, Jianxiang Feng, Matthias Humt, Marcus Gerhard Müller, and Rudolph Triebel. Trust your robots! Predictive uncertainty estimation of neural networks with sparse Gaussian processes. In *Conference on Robot Learning*, pages 1168–1179. PMLR, 2022.
- [1003] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019.
- [1004] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

- [1005] Kwonjoon Lee, Subhansu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [1006] Kyeongwon Lee and Jaeyong Lee. Asymptotic properties for Bayesian neural network in besov space. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5641–5653. Curran Associates, Inc., 2022.
- [1007] Seunghyun Lee and Yuqi Gu. Deep discrete encoders: Identifiable deep generative models for rich data with discrete latent layers, 2025.
- [1008] W.B. Leeds, C.K. Wikle, and J. Fiechter. Emulator-assisted reduced-rank ecological data assimilation for nonlinear multivariate dynamical spatio-temporal processes. *Statistical Methodology*, 17:126–138, 2014. Modern Statistical Methods in Ecology.
- [1009] Sirio Legramanti, Daniele Durante, and David B Dunson. Bayesian cumulative shrinkage for infinite factorizations. *Biometrika*, 107(3):745–752, 2020.
- [1010] Nils Lehmann, Nina Maria Gottschling, Jakob Gawlikowski, Adam J. Stewart, Stefan Depeweg, and Eric Nalisnick. Lightning uq box: Uncertainty quantification for neural networks. *Journal of Machine Learning Research*, 26(54):1–7, 2025.
- [1011] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*. Cambridge university press, 2005.
- [1012] Pablo Lemos, Liam Parker, ChangHoon Hahn, Shirley Ho, Michael Eickenberg, Jiamin Hou, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Bruno Régalo-Saint Blancard, and David Spergel. Field-level simulation-based inference of galaxy clustering with convolutional neural networks. *Phys. Rev. D*, 109: 083536, 2024.
- [1013] Peter Lenk and Bryan Orme. The value of informative priors in Bayesian inference with sparse data. *Journal of Marketing Research*, 46(6):832–845, 2009.
- [1014] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.
- [1015] Guy Lever, François Laviolette, and John Shawe-Taylor. Distribution-dependent pac-bayes priors. In Marcus Hutter, Frank Stephan, Vladimir Vovk, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 119–133, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [1016] Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013. Special Issue on Algorithmic Learning Theory.
- [1017] Isaac Levi. *The Enterprise of Knowledge: an Essay on Knowledge, Credal Probability, and Chance*. MIT Press, 1980.
- [1018] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pages 148–156. Morgan Kaufmann, 1994.
- [1019] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In Bruce W. Croft and C. J. van Rijsbergen, editors, *SIGIR '94*, pages 3–12, London, 1994. Springer London.
- [1020] Aodong Li, Alex Boyd, Padhraic Smyth, and Stephan Mandt. Detecting and adapting to irregular distribution shifts in Bayesian online learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6816–6828. Curran Associates, Inc., 2021.

- [1021] Bai Li, Changyou Chen, Hao Liu, and Lawrence Carin. On connecting stochastic gradient MCMC and differential privacy. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 557–566. PMLR, 2019.
- [1022] Bo Li, Yifei Shen, Jingkan Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. Sparse mixture-of-experts are domain generalizable learners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [1023] Bolian Li and Ruqi Zhang. Entropy-MCMC: Sampling from flat basins with ease. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 19556–19579, 2024.
- [1024] Cheuk Ting Li. Channel simulation: Theory and applications to lossy compression and differential privacy. *Foundations and Trends in Communications and Information Theory*, 21(6):847–1106, 2024.
- [1025] Cheuk Ting Li and Abbas El Gamal. Strong functional representation lemma and applications to coding theorems. *IEEE Transactions on Information Theory*, 64(11):6967–6978, 2018.
- [1026] Fengyi Li, Ricardo Baptista, and Youssef Marzouk. Expected information gain estimation via density approximations: Sample allocation and dimension reduction, 2024.
- [1027] Fuxin Li, Catalin Ionescu, and Cristian Sminchisescu. Random Fourier approximations for skewed multiplicative histogram kernels. In Michael Goesele, Stefan Roth, Arjan Kuijper, Bernt Schiele, and Konrad Schindler, editors, *Pattern Recognition*, pages 262–271, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [1028] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1029] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1030] Lingxiao Li, Qiang Liu, Anna Korba, Mikhail Yurochkin, and Justin Solomon. Sampling with mollified interaction energy descent, 2022.
- [1031] Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*. Springer, third edition edition, 2008.
- [1032] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression, 2017.
- [1033] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 4328–4343. Curran Associates, Inc., 2022.
- [1034] Yikuan Li, Shishir Rao, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Gholamreza Salimi-Khorshidi, Mohammad Mamouei, Thomas Lukasiewicz, and Kazem Rahimi. Deep Bayesian Gaussian processes for uncertainty estimation in electronic health records. *Scientific reports*, 11(1):20685, 2021.
- [1035] Yingzhen Li and Richard Turner. Rényi divergence variational inference. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1036] Yucen Li, Tim G. J. Rudner, and Andrew Gordon Wilson. A study of Bayesian neural network surrogates for Bayesian optimization. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 47003–47041, 2024.
- [1037] Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, L. Fei-Fei, Matei Zaharia, Ce Zhang, and James Zou. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, 4(8):669–677, 2022.

- [1038] Xinzhu Liang, Joseph M. Lukens, Sanjaya Lohani, Brian T. Kirby, Thomas A. Searles, Xin Qiu, and Kody J. H. Law. Scalable Bayesian Monte Carlo: fast uncertainty estimation beyond deep ensembles, 2025.
- [1039] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training, 2018.
- [1040] Vincent Lim, Ellen Novoseller, Jeffrey Ichnowski, Huang Huang, and Ken Goldberg. Policy-based Bayesian experimental design for non-differentiable implicit models, 2022.
- [1041] Jihao Andreas Lin, Javier Antoran, and José Miguel Hernández-Lobato. Online Laplace model selection revisited. In *Symposium on Advances in Approximate Bayesian Inference (AABI)*, volume 5, 2023.
- [1042] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, 2022. Association for Computational Linguistics.
- [1043] Vivian Lin, Kuk Jin Jang, Souradeep Dutta, Michele Caprio, Oleg Sokolsky, and Insup Lee. DC4L: Distribution shift recovery via data-driven control for deep learning models. In Alessandro Abate, Mark Cannon, Kostas Margellos, and Antonis Papachristodoulou, editors, *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pages 1526–1538. PMLR, 2024.
- [1044] Wu Lin. Invariance of Natural-Gradients. <https://yorkerlin.github.io/posts/2021/11/Geomopt03/>. Accessed: 2025-27-10.
- [1045] Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3992–4002. PMLR, 2019.
- [1046] Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Stein’s lemma for the reparameterization trick with exponential family mixtures, 2019.
- [1047] Wu Lin, Mark Schmidt, and Mohammad Emtiyaz Khan. Handling the positive-definite constraint in the Bayesian learning rule. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6116–6126. PMLR, 2020.
- [1048] Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E. Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? A second-order perspective. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 29949–29973. PMLR, 2024.
- [1049] D. V. Lindley. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986 – 1005, 1956.
- [1050] D. V. Lindley. *Bayesian Statistics: a Review*. Society for Industrial and Applied Mathematics, 1972.
- [1051] D. V. Lindley, A. Tversky, and R. V. Brown. On the reconciliation of probability assessments. *Journal of the Royal Statistical Society: Series A (General)*, 142(2):146–162, 1979.
- [1052] Julia Linhart, Alexandre Gramfort, and Pedro Rodrigues. L-C2ST: Local diagnostics for posterior approximations in simulation-based inference. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 56384–56410. Curran Associates, Inc., 2023.
- [1053] Julia Linhart, Gabriel Cardoso, Alexandre Gramfort, Sylvain Le Corff, and Pedro L. C. Rodrigues. Diffusion posterior sampling for simulation-based inference in tall data settings. *Transactions on Machine Learning Research*, 2026. J2C Certification.

- [1054] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2022.
- [1055] Maximilian Lipp, Benjamin Kurt Miller, Lyubov Amitonova, and Patrick Forré. Generalizing coverage plots for simulation-based inference. *Transactions on Machine Learning Research*, 2026.
- [1056] Phillip Lippe, Taco Cohen, and Efstratios Gavves. Efficient neural causal discovery without acyclicity constraints, 2021.
- [1057] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, third edition edition, 2019.
- [1058] Chang Liu and Jun Zhu. Riemannian Stein variational gradient descent for Bayesian inference. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3627–3634. AAAI Press, 2018.
- [1059] Feng Liu, Wenkai Xu, Jie Lu, and Danica J. Sutherland. Meta two-sample testing: Learning kernels for testing with limited data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5848–5860. Curran Associates, Inc., 2021.
- [1060] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search, 2018.
- [1061] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7498–7512. Curran Associates, Inc., 2020.
- [1062] Jeremiah Zhe Liu, Shreyas Padhy, Jie Ren, Zi Lin, Yeming Wen, Ghassen Jerfel, Zachary Nado, Jasper Snoek, Dustin Tran, and Balaji Lakshminarayanan. A simple approach to improve single-model deep uncertainty via distance-awareness. *Journal of Machine Learning Research*, 24(42):1–63, 2023.
- [1063] Qiang Liu. Stein variational gradient descent as gradient flow. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1064] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1065] Qiang Liu and Dilin Wang. Stein variational gradient descent as moment matching. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1066] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 276–284, New York, New York, USA, 2016. PMLR.
- [1067] Tianle Liu, Promit Ghosal, Krishnakumar Balasubramanian, and Natesh Pillai. Towards understanding the dynamics of Gaussian-Stein variational gradient descent. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 61234–61291. Curran Associates, Inc., 2023.
- [1068] Xing Liu, Harrison Zhu, Jean-François Ton, George Wynne, and Andrew Duncan. Grassmann Stein variational gradient descent, 2022.
- [1069] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18261–18271. Curran Associates, Inc., 2020.

- [1070] Ziming Liu, Surya Ganguli, and Andreas S. Tolias. From kepler to Newton: Inductive biases guide learned world models in transformers. In *Forty-third International Conference on Machine Learning*, 2026.
- [1071] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [1072] Philip M. Long and Rocco A. Servedio. Restricted boltzmann machines are hard to approximately evaluate or simulate. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 703–710. Omnipress, 2010.
- [1073] Quan Long, Marco Scavino, Raúl Tempone, and Suojin Wang. Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations. *Computer Methods in Applied Mechanics and Engineering*, 2013.
- [1074] Romain Lopez, Jeffrey Regier, Michael B. Cole, Michael I. Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018.
- [1075] Lopez-Rey, Carot-Sans, Ouchi, Torres, and Pontes. Use of Bayesian approaches in oncology clinical trials: a cross-sectional analysis. *Frontiers in Pharmacology*, 2025.
- [1076] Antonio Loquercio, Mattia Segù, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.
- [1077] Lars Lorch, Jonas Rothfuss, Bernhard Schölkopf, and Andreas Krause. Dibs: Differentiable Bayesian structure learning, 2021.
- [1078] Gabriel J Lord, Catherine E Powell, and Tony Shardlow. *An introduction to computational stochastic PDEs*. Cambridge University Press, 2014.
- [1079] Marco Lorenzi and Maurizio Filippone. Constraining the dynamics of deep probabilistic models. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3227–3236. PMLR, 2018.
- [1080] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1540–1552. PMLR, 2020.
- [1081] Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14223–14247. PMLR, 2022.
- [1082] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32819–32848. PMLR, 2024.
- [1083] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1708–1716, New York, New York, USA, 2016. PMLR.
- [1084] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An End-To-End Deep Video Compression Framework . In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10998–11007, Los Alamitos, CA, USA, 2019. IEEE Computer Society.
- [1085] Jianfeng Lu, Yulong Lu, and James Nolen. Scaling limit of the Stein variational gradient descent: The mean field regime. *SIAM Journal on Mathematical Analysis*, 51(2):648–671, 2019.

- [1086] Yingzhou Lu, Tianyi Chen, Nan Hao, Capucine Van Rechem, Jintai Chen, and Tianfan Fu. Uncertainty quantification and interpretability for clinical trial approval prediction. *Health Data Science*, 4:0126, 2024.
- [1087] Yulong Lu, Jianfeng Lu, and James Nolen. Accelerating Langevin sampling with birth-death, 2019.
- [1088] Jan-Matthis Lückmann. *Simulation-based inference for neuroscience and beyond*. PhD thesis, Dissertation, Tübingen, Universität Tübingen, 2022, 2021.
- [1089] Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebia, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 33270–33288. PMLR, 2024.
- [1090] Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1091] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks, 2018.
- [1092] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 343–351. PMLR, 2021.
- [1093] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2952–2960, New York, New York, USA, 2016. PMLR.
- [1094] Mantas Lukoševičius. *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [1095] Yueming Lyu and Ivor W. Tsang. Black-box optimizer with stochastic implicit natural gradient. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 217–232, Cham, 2021. Springer International Publishing.
- [1096] Chao Ma and José Miguel Hernández-Lobato. Functional variational inference based on stochastic process generators. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21795–21807. Curran Associates, Inc., 2021.
- [1097] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient MCMC. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [1098] Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019.
- [1099] Yuchen Ma, Dennis Frauen, Emil Javurek, and Stefan Feuerriegel. Foundation models for causal inference via prior-data fitted networks, 2025.
- [1100] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.
- [1101] Jacob Reinier Maat, Nikos Gianniotis, and Pavlos Protopapas. Efficient optimization of echo state networks for time series datasets. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018.

- [1102] Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7 (1):83–122, 1975.
- [1103] Steven N MacEachern. Decision theoretic aspects of dependent nonparametric processes. *Bayesian methods with applications to science, policy and official statistics*, pages 551–560, 2001.
- [1104] David MacKay. Bayesian model comparison and backprop nets. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.
- [1105] David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
- [1106] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4 (5):720–736, 1992.
- [1107] David J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.
- [1108] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.
- [1109] David J. C. MacKay. Choice of basis for Laplace approximation. *Machine Learning*, 33(1):77–86, 1998.
- [1110] David J. C. MacKay. Frequently asked questions about Bayesian methods for neural networks. http://inference.org.uk/mackay/Bayes_FAQ.html, 2004. Inference Group, University of Cambridge.
- [1111] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [1112] David John Cameron Mackay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.
- [1113] David John Cameron MacKay. Introduction to Gaussian processes. 1998.
- [1114] Chris Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1115] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [1116] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1117] Wesley J. Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited, 2020.
- [1118] Wesley J Maddox, Samuel Stanton, and Andrew G Wilson. Conditioning sparse variational Gaussian processes for online decision-making. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6365–6379. Curran Associates, Inc., 2021.
- [1119] David Madras, Toni Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1120] Seiji Maekawa, Dan Zhang, Hannah Kim, Sajjadur Rahman, and Estevam Hruschka. Low-resource interactive active labeling for fine-tuning language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3230–3242, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

- [1121] Jay Magidson and Jeroen K Vermunt. Latent class models. *The SAGE Handbook of Quantitative Methodology for the Social Sciences*, 10:175–198, 2004.
- [1122] Kaitlin Maile, Emmanuel Rachelson, Hervé Luga, and Dennis George Wilson. When, where, and how to add new neurons to anns. In Isabelle Guyon, Marius Lindauer, Mihaela van der Schaar, Frank Hutter, and Roman Garnett, editors, *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pages 18/1–12. PMLR, 2022.
- [1123] Enes Makalic and Daniel F. Schmidt. A simple sampler for the horseshoe estimator. *IEEE Signal Processing Letters*, 23(1):179–182, 2016.
- [1124] Hugo Monzón Maldonado, Nico Daheim, Thomas Möllenhoff, Iryna Gurevych, and Mohammad Emtiyaz Khan. Variational model merging for Pareto front estimation in multitask finetuning, 2024.
- [1125] Andrey Malinin and Mark J. F. Gales. Uncertainty estimation in autoregressive structured prediction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [1126] V. Mallet, D.E. Keyes, and F.E. Fendell. Modeling wildland fire propagation with level set methods. *Computers & Mathematics with Applications*, 57(7):1089–1101, 2009.
- [1127] Shireen Kudukkil Manchingal and Fabio Cuzzolin. Epistemic deep learning, 2022.
- [1128] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.
- [1129] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. PEFT: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>, 2022.
- [1130] Katerina Margatina, Loic Barrault, and Nikolaos Aletras. On the importance of effectively adapting pre-trained language models for active learning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 825–836, Dublin, Ireland, 2022. Association for Computational Linguistics.
- [1131] Charles C. Margossian, Matthew D. Hoffman, Pavel Sountsov, Lionel Riou-Durand, Aki Vehtari, and Andrew Gelman. Nested $\hat{R}$: Assessing the Convergence of Markov Chain Monte Carlo When Running Many Short Chains. *Bayesian Analysis*, 20(4):1587 – 1614, 2025.
- [1132] Jean-Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.
- [1133] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3403–3412. PMLR, 2018.
- [1134] Ilia Markov, Adrian Vladu, Qi Guo, and Dan Alistarh. Quantized distributed training of large models with convergence guarantees. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24020–24044. PMLR, 2023.
- [1135] James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.
- [1136] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2408–2417, Lille, France, 2015. PMLR.
- [1137] Gloria Martín. Range encoding: an algorithm for removing redundancy from a digitised message. In *Video and Data Recording Conference, Southampton, 1979*, pages 24–27, 1979.
- [1138] Osvaldo Martin, Ravin Kumar, Junpeng Lao, and et al. PyMC: A modern and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9:e1419, 2023.

- [1139] Pedro Henrique Martins, Jo ao Alves, Patrick Fernandes, Nuno M. Guerreiro, Ricardo Rei, Amin Farajian, Mateusz Klimaszewski, Duarte M. Alves, José Pombal, Nicolas Boizard, Manuel Faysse, Pierre Colombo, François Yvon, Barry Haddow, José G. C. de Souza, Alexandra Birch, and André F. T. Martins. Eurollm-9b: Technical report, 2025.
- [1140] K. Marton. A measure concentration inequality for contracting markov chains. *Geometric & Functional Analysis GAFA*, 6(3):556–571, 1996.
- [1141] Katalin Marton. A measure concentration inequality for contracting Markov chains Erratum. *Geometric and Functional Analysis*, 7(3), 1997.
- [1142] Andres Masegosa, Stephan Lorenzen, Christian Igel, and Yevgeny Seldin. Second order PAC-Bayesian bounds for the weighted majority vote. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5263–5273. Curran Associates, Inc., 2020.
- [1143] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.
- [1144] Michael S Matena and Colin Raffel. Merging models with Fisher-weighted averaging. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17703–17716. Curran Associates, Inc., 2022.
- [1145] Takuo Matsubara, Chris J. Oates, and François-Xavier Briol. The ridgelet prior: A covariance function approach to prior specification for Bayesian neural networks. *Journal of Machine Learning Research*, 22(157):1–57, 2021.
- [1146] Pierre-Alexandre Mattei and Jes Frellsen. Leveraging the exact likelihood of deep latent variable models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1147] Pierre-Alexandre Mattei and Jes Frellsen. MIWAE: Deep generative modelling and imputation of incomplete data sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4413–4423. PMLR, 2019.
- [1148] Alexander G. de G. Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 231–239, Cadiz, Spain, 2016. PMLR.
- [1149] Alexander Graeme de Garis Matthews. *Scalable Gaussian Process Inference Using Variational Methods*. PhD thesis, University of Cambridge, 2017.
- [1150] Filip Matzner. Hyperparameter tuning in echo state networks. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22*, pages 404–412, New York, NY, USA, 2022. Association for Computing Machinery.
- [1151] Andreas Maurer. A note on the PAC Bayesian theorem, 2004.
- [1152] Peter S. Maybeck. *Stochastic models, estimation, and control*. Academic Press, 1979.
- [1153] Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzębski. Molecule attention transformer, 2020.
- [1154] David A. McAllester. Some PAC-Bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98*, pages 230–234, New York, NY, USA, 1998. Association for Computing Machinery.
- [1155] David A. McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003.

- [1156] Andrew Kachites McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, Madison, Wisconsin, USA, July 24-27, 1998, pages 350–358. Morgan Kaufmann, 1998.
- [1157] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
- [1158] C.K. McDermott, P.L. and Wikle. An ensemble quadratic echo state network for non-linear spatio-temporal forecasting. *Stat*, 6:315–330, 2017. <https://doi.org/10.1002/sta4.160>.
- [1159] P.L. McDermott and C. K. Wikle. Deep echo state networks with uncertainty quantification for spatio-temporal forecasting. *Environmetrics*, 30:e2553, 2019. <https://doi.org/10.1002/env.2553>.
- [1160] Brockway McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4):115–116, 1956.
- [1161] Mauricius Munhoz de Medeiros, Norberto Hoppen, and Antonio Carlos Gastaud Maçada. Data science for business: benefits, challenges and opportunities. *The Bottom Line*, 33(2):149–163, 2020.
- [1162] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-Sinkhorn networks, 2018.
- [1163] Jorge M. Mendes, Patrícia Cortés de Zea Bermudez, José Pereira, K. F. Turkman, and M. J. P. Vasconcelos. Spatial extremes of wildfire sizes: Bayesian hierarchical models for extremes. *Environmental and Ecological Statistics*, 17(1):1–28, 2010.
- [1164] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 2019.
- [1165] Qiuxian Meng and Yongyou Zhang. Amortized variational inference for deep Gaussian processes, 2024.
- [1166] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.
- [1167] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [1168] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11913–11924. Curran Associates, Inc., 2020.
- [1169] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- [1170] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA, 2016. PMLR.
- [1171] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2017.
- [1172] Jacques Paul Migne. De coelesti hierarchia. In *Patrologia Graeca*, volume 3. Paris : Imprimerie Catholique, 1857.

- [1173] Petrus Mikkola, Osvaldo A. Martin, Suyog Chandramouli, Marcelo Hartmann, Oriol Abril Pla, Owen Thomas, Henri Pesonen, Jukka Corander, Aki Vehtari, Samuel Kaski, Paul-Christian Bürkner, and Arto Klami. Prior Knowledge Elicitation: The Past, Present, and Future. *Bayesian Analysis*, 19(4):1129 – 1161, 2024.
- [1174] Benjamin K Miller, Alex Cole, Patrick Forré, Gilles Louppe, and Christoph Weniger. Truncated marginal neural ratio estimation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 129–143. Curran Associates, Inc., 2021.
- [1175] Benjamin K Miller, Christoph Weniger, and Patrick Forré. Contrastive neural ratio estimation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 3262–3278. Curran Associates, Inc., 2022.
- [1176] Benjamin Kurt Miller, Alex Cole, Christoph Weniger, Francesco Nattino, Ou Ku, and Meiert W. Grootes. swyft: Truncated marginal neural ratio estimation in python. *Journal of Open Source Software*, 7(75):4205, 2022.
- [1177] Benjamin Kurt Miller, Marco Federici, Christoph Weniger, and Patrick Forré. Simulation-based inference with the generalized Kullback-Leibler divergence, 2023.
- [1178] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.
- [1179] Edward Milsom, Ben Anson, and Laurence Aitchison. Convolutional deep kernel machines. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 1036–1064, 2024.
- [1180] Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Hölten, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15630–15649. PMLR, 2022.
- [1181] Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In Jack S. Breese and Daphne Koller, editors, *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 362–369. Morgan Kaufmann, 2001.
- [1182] Thomas P Minka. Bayesian model averaging is not model combination, 2002. Available electronically at <https://tminka.github.io/papers/minka-bma-isnt-mc.pdf>.
- [1183] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1184] Adrian Robert Minut, Nico Daheim, Marco Miani, Mohammad Emtiyaz Khan, Wu Lin, and Thomas Möllenhoff. Soap-bubbles: Structured weight uncertainty for neural networks, 2026.
- [1185] Enrique Miranda and Ignacio Montes. Centroids of the core of exact capacities: a comparative study. *Annals of Operations Research*, 321:409–449, 2023.
- [1186] Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. Slang: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1187] Aayush Mishra, Daniel Habermann, Marvin Schmitt, Stefan T. Radev, and Paul-Christian Bürkner. Robust amortized Bayesian inference with self-consistency losses on unlabeled data, 2025.

- [1188] Siddharth Mishra-Sharma and Kyle Cranmer. Neural simulation-based inference approach for characterizing the galactic center γ -ray excess. *Phys. Rev. D*, 105:063017, 2022.
- [1189] Rupert Mitchell, Robin Menzenbach, Kristian Kersting, and Martin Mundt. Self-expanding neural networks, 2023.
- [1190] T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.
- [1191] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation, 2016.
- [1192] Bruno Mlodozienec, David Krueger, and Richard Turner. Implicitly Bayesian prediction rules in deep learning. In Javier Antorán and Christian A. Naesseth, editors, *Proceedings of the 6th Symposium on Advances in Approximate Bayesian Inference*, volume 253 of *Proceedings of Machine Learning Research*, pages 79–110. PMLR, 2024.
- [1193] Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. Sample complexity of reinforcement learning using linearly combined model ensembles. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2010–2020. PMLR, 2020.
- [1194] Salman Mohamadi, Gianfranco Doretto, and Donald A. Adjeroh. Deep active ensemble sampling for image classification. In Lei Wang, Juergen Gall, Tat-Jun Chin, Imari Sato, and Rama Chellappa, editors, *Computer Vision – ACCV 2022*, pages 713–729, Cham, 2023. Springer Nature Switzerland.
- [1195] Théo Moins, Julyan Arbel, Anne Dufloy, and Stéphane Girard. On the Use of a Local $\hat{R}$ to Improve MCMC Convergence Diagnostic. *Bayesian Analysis*, 20(1):131 – 156, 2025.
- [1196] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2498–2507. PMLR, 2017.
- [1197] Thomas Möllenhoff and Mohammad Emtiyaz Khan. SAM as an optimal relaxation of Bayes. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [1198] Tom Monnier, Thibault Groueix, and Mathieu Aubry. Deep transformation-invariant clustering. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7945–7955. Curran Associates, Inc., 2020.
- [1199] Andrea Montanari. Sampling, diffusions, and stochastic localization. 2025.
- [1200] Philip Mortimer, Cristiana Diaconu, Tommy Rochussen, Bruno Kacper Mlodozienec, and Richard E. Turner. Incremental transformer neural processes. In *Forty-third International Conference on Machine Learning*, 2026.
- [1201] Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, 2020.
- [1202] Michael C Mozer and Jonathan Bachrach. Discovering the structure of a reactive environment by exploration. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.
- [1203] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision, 2019.
- [1204] Bálint Mucsányi, Michael Kirchhof, and Seong Joon Oh. Benchmarking uncertainty disentanglement: Specialized uncertainties for specialized tasks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 50972–51038. Curran Associates, Inc., 2024.
- [1205] Bálint Mucsányi, Nathaël Da Costa, and Philipp Hennig. Rethinking approximate Gaussian inference in classification, 2025.

- [1206] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H.S. Torr, and Yarin Gal. Deep deterministic uncertainty: A new simple baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24384–24394, 2023.
- [1207] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.
- [1208] Samuel Müller, Noah Hollmann, Sebastian Pineda-Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [1209] Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25444–25470. PMLR, 2023.
- [1210] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.
- [1211] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [1212] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.
- [1213] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.
- [1214] Iain Murray and Zoubin Ghahramani. A note on the evidence and Bayesian Occam’s razor. Technical Report GCNU-TR 2005-003, Gatsby Computational Neuroscience Unit, University College London, 2005.
- [1215] Jared S. Murray. Multiple Imputation: A Review of Practical and Theoretical Findings. *Statistical Science*, 33(2):142 – 159, 2018.
- [1216] Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural PDE solvers. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 38777–38825, 2025.
- [1217] Ion Muslea, Steven Minton, and Craig A. Knoblock. Active + semi-supervised learning = robust multi-view learning. In Claude Sammut and Achim G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 435–442. Morgan Kaufmann, 2002.
- [1218] Stephen Mussmann, Julia Reisler, Daniel Tsai, Ehsan Mousavi, Shayne O’Brien, and Moises Goldszmidt. Active learning with expected error reduction, 2022.
- [1219] Jay I. Myung, Daniel R. Cavagnaro, and Mark A. Pitt. A tutorial on adaptive design optimization. *Journal of Mathematical Psychology*, 57(3):53–67, 2013.
- [1220] Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael W. Dusenberry, Sebastian Farquhar, Qixuan Feng, Angelos Filos, Marton Havasi, Rodolphe Jenatton, Ghassen Jerfel, Jeremiah Liu, Zeld Mariet, Jeremy Nixon, Shreyas Padhy, Jie Ren, Tim G. J. Rudner, Faris Sbahi, Yeming Wen, Florian Wenzel, Kevin Murphy, D. Sculley, Balaji Lakshminarayanan, Jasper Snoek, Yarin Gal, and Dustin Tran. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning, 2021.
- [1221] Shinichi Nakajima and Sumio Watanabe. Variational Bayes solution of linear neural networks and its generalization performance. *Neural Computation*, 19(4):1112–1153, 2007.
- [1222] Eric T. Nalisnick and Padhraic Smyth. Learning approximately objective priors. In Gal Elidan, Kristian Kersting, and Alexander Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.
- [1223] Eric Thomas Nalisnick. *On priors for Bayesian neural networks*. PhD thesis, University of California, Irvine, 2018.

- [1224] Harikrishna Narasimhan, Wittawat Jitkrittum, Aditya K Menon, Ankit Rawat, and Sanjiv Kumar. Post-hoc estimators for learning to defer to an expert. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 29292–29304. Curran Associates, Inc., 2022.
- [1225] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, 107:107501, 2020.
- [1226] Radford M Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.
- [1227] Radford M. Neal. *Priors for Infinite Networks*, pages 29–53. Springer New York, New York, NY, 1996.
- [1228] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.
- [1229] Radford M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 113–162. CRC, 2011.
- [1230] Marcos Negre Saura, Richard Allmendinger, Wei Pan, and Theodore Papamarkou. Spatial-aware decision-making with ring attractors in reinforcement learning systems. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 164910–164942. Curran Associates, Inc., 2025.
- [1231] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Structured Bayesian pruning via log-normal multiplicative noise. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1232] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [1233] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.
- [1234] Andrew Y. Ng. Preventing "overfitting" of cross-validation data. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, 1997.
- [1235] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.
- [1236] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.
- [1237] Hieu Tat Nguyen and Arnold W. M. Smeulders. Active learning using pre-clustering. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.
- [1238] Huy Nguyen, TrungTin Nguyen, and Nhat Ho. Demystifying softmax gating function in Gaussian mixture of experts. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 4624–4652. Curran Associates, Inc., 2023.
- [1239] Huy Nguyen, Pedram Akbarian, Trungtin Nguyen, and Nhat Ho. A general theory for softmax gating multinomial logistic mixture of experts. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 37617–37648. PMLR, 2024.
- [1240] Huy Nguyen, Pedram Akbarian Saravi, Fanqi Yan, and Nhat Ho. Statistical perspective of top-k sparse softmax gating mixture of experts. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 3614–3651, 2024.

- [1241] Huy Nguyen, Xing Han, Carl Harris, Suchi Saria, and Nhat Ho. On expert estimation in hierarchical mixture of experts: Beyond softmax gating functions, 2024.
- [1242] Huy Nguyen, Pedram Akbarian Saravi, Trang Pham, Thien Trang Nguyen Vu, Shujian Zhang, and Nhat Ho. Statistical advantages of perturbing cosine router in mixture of experts. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 95812–95844, 2025.
- [1243] Huy Nguyen, Thong T. Doan, Quang Pham, Nghi D. Q. Bui, Nhat Ho, and Alessandro Rinaldo. On deepseekmoe: Statistical benefits of shared experts and normalized sigmoid gating, 2025.
- [1244] Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, Kenji Satou, et al. DNA sequence classification by convolutional neural network. *Journal of Biomedical Science and Engineering*, 9(05):280, 2016.
- [1245] Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16569–16594. PMLR, 2022.
- [1246] Lorenzo Noci, Gregor Bachmann, Kevin Roth, Sebastian Nowozin, and Thomas Hofmann. Precise characterization of the prior predictive distribution of deep ReLU networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20851–20862. Curran Associates, Inc., 2021.
- [1247] Emmy Noether. *Invariante Variationsprobleme*, pages 231–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.
- [1248] Anna Norberg, Nerea Abrego, F Guillaume Blanchet, Frederick R Adler, Barbara J Anderson, Jani Anttila, Miguel B Araújo, Tad Dallas, David Dunson, Jane Elith, et al. A comprehensive evaluation of predictive performance of 33 species distribution models at species and community levels. *Ecological Monographs*, 89(3):e01370, 2019.
- [1249] William D. Nordhaus. Two centuries of productivity growth in computing. *The Journal of Economic History*, 67(1):128–159, 2007.
- [1250] Sebastian Ober and Laurence Aitchison. A variational approximate posterior for the deep wishart process. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6567–6579. Curran Associates, Inc., 2021.
- [1251] Sebastian W Ober and Laurence Aitchison. Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8248–8259. PMLR, 2021.
- [1252] Sebastian W. Ober and Carl E. Rasmussen. Benchmarking the neural linear model for regression. In *Second Symposium on Advances in Approximate Bayesian Inference*, 2019.
- [1253] Sebastian W. Ober, Carl E. Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1206–1216. PMLR, 2021.
- [1254] Sebastian W. Ober, Ben Anson, Edward Milsom, and Laurence Aitchison. An improved variational approximate posterior for the deep Wishart process. In Robin J. Evans and Ilya Shpitser, editors, *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 1555–1563. PMLR, 2023.
- [1255] A. O’Hagan. Eliciting expert beliefs in substantial practical applications. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1):21–35, 1998.
- [1256] Deniz Oktay, Johannes Ballé, Saurabh Singh, and Abhinav Shrivastava. Scalable model compression by entropy penalized reparameterization, 2019.

- [1257] James Oldfield, Markos Georgopoulos, Grigorios G. Chrysos, Christos Tzelepis, Yannis Panagakis, Mihalis Nicolaou, Jiankang Deng, and Ioannis Patras. Multilinear mixture of experts: Scalable expert specialization through factorization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 53022–53063. Curran Associates, Inc., 2024.
- [1258] Yann Ollivier. Online natural gradient as a Kalman filter. *Electronic Journal of Statistics*, 12(2):2930 – 2961, 2018.
- [1259] Miles Olson, Elizabeth Santorella, Louis C. Tiao, Sait Cakmak, David Eriksson, Mia Garrard, Sam Daulton, Maximilian Balandat, Eytan Bakshy, Elena Kashtelyan, Zhiyuan Jerry Lin, Sebastian Ament, Bernard Beckerman, Eric Onofrey, Paschal Igusti, Cristian Lara, Benjamin Letham, Cesar Cardoso, Shiyun Sunny Shen, Andy Chenyuan Lin, and Matthew Grange. Ax: A platform for adaptive experimentation. In *AutoML 2025 ABCD Track*, 2025.
- [1260] Emre Onal, Klemens Flöge, Emma Caldwell, Arsen Sheverdin, and Vincent Fortuin. Gaussian stochastic weight averaging for Bayesian low-rank adaptation of large language models, 2024.
- [1261] Victor M-H Ong, David J Nott, and Michael S Smith. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3):465–478, 2018.
- [1262] OpenAI. GPT-4 technical report, 2023.
- [1263] Thomas Opitz. Eva 2021 data challenge on spatiotemporal prediction of wildfire extremes in the usa. *Extremes*, 26(2):241–250, 2023.
- [1264] Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009.
- [1265] Luis A. Ortega, Simón Rodríguez Santana, and Daniel Hernández-Lobato. Variational linearized Laplace approximation for Bayesian deep learning, 2023.
- [1266] Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz Khan, Anirudh Jain, Runa Eschenhagen, Richard Turner, and Rio Yokota. Practical deep learning with Bayesian principles. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1267] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2701–2710. PMLR, 2017.
- [1268] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [1269] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1270] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1271] Ian Osband, Seyed Mohammad Asghari, Benjamin Van Roy, Nat McAleese, John Aslanides, and Geoffrey Irving. Fine-tuning language models via epistemic neural networks, 2023.
- [1272] Felix Otto. The geometry of dissipative evolution equations: The porous medium equation. *Communications in Partial Differential Equations*, 26(1-2):101–174, 2001.

- [1273] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.
- [1274] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1275] Otso Ovaskainen and Nerea Abrego. *Joint Species Distribution Modelling: With Applications in R*. Cambridge University Press, 2020.
- [1276] Otso Ovaskainen, David B Roy, Richard Fox, and Barbara J Anderson. Uncovering hidden spatial structure in species communities with spatially explicit joint species distribution models. *Methods in Ecology and Evolution*, 7(4):428–436, 2016.
- [1277] Otso Ovaskainen, Gleb Tikhonov, David Dunson, Vidar Grøtan, Steinar Engen, Bernt-Erik Sæther, and Nerea Abrego. How are species interactions structured in species-rich communities? a new method for analysing time-series data. *Proceedings of the Royal Society B: Biological Sciences*, 284(1855):20170768, 2017.
- [1278] Otso Ovaskainen, Gleb Tikhonov, Anna Norberg, F. Guillaume Blanchet, Leo Duan, David Dunson, Tomas Roslin, and Nerea Abrego. How to make more out of community data? a conceptual framework and its implementation as models and software. *Ecology Letters*, 20(5):561–576, 2017.
- [1279] Otso Ovaskainen, Steven Winter, Gleb Tikhonov, Nerea Abrego, Sten Anslan, Jeremy R. deWaard, Stephanie L. deWaard, Brian L. Fisher, Brendan Furneaux, Bess Hardwick, Deirdre Kerdraon, Mikko Pentinsaari, Dimby Raharinjanahary, Eric Tsiriniaina Rajoelison, Sujeevan Ratnasingham, Panu Somervuo, Jayme E. Sones, Evgeny V. Zakharov, Paul D. N. Hebert, Tomas Roslin, and David Dunson. Common to rare transfer learning (coral) enables inference and prediction for a quarter million rare malagasy arthropods. *bioRxiv*, 2025.
- [1280] Antony Overstall and James McGree. Bayesian Decision-Theoretic Design of Experiments Under an Alternative Model. *Bayesian Analysis*, 17(4):1021 – 1041, 2022.
- [1281] R Pacelli, S Ariosto, Mauro Pastore, F Ginelli, Marco Gherardi, and Pietro Rotondo. A statistical mechanics framework for Bayesian deep neural networks beyond the infinite-width limit. *Nature Machine Intelligence*, 5(12):1497–1507, 2023.
- [1282] Brooks Paige, Frank Wood, Arnaud Doucet, and Yee Whye Teh. Asynchronous anytime sequential Monte Carlo. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [1283] John Paisley, Chong Wang, David M. Blei, and Michael I. Jordan. Nested hierarchical Dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):256–270, 2015.
- [1284] Avik Pal. Lux: Explicit parameterization of deep neural networks in julia, 2023.
- [1285] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1595–1605. PMLR, 2020.
- [1286] Deep Shankar Pandey, Spandan Pyakurel, and Qi Yu. Be confident in what you know: Bayesian parameter efficient fine-tuning of vision foundation models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 44814–44844. Curran Associates, Inc., 2024.

- [1287] George Papamakarios. Neural density estimation and likelihood-free inference, 2019.
- [1288] George Papamakarios and Iain Murray. Fast ϵ -free inference of simulation models with Bayesian conditional density estimation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1289] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1290] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2019.
- [1291] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 837–848. PMLR, 2019.
- [1292] Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, Aliaksandr Hubin, Alexander Immer, Theofanis Karaletsos, Mohammad Emtiyaz Khan, Agustinus Kristiadi, Yingzhen Li, Stephan Mandt, Christopher Nemeth, Michael A Osborne, Tim G. J. Rudner, David Rügamer, Yee Whye Teh, Max Welling, Andrew Gordon Wilson, and Ruqi Zhang. Position: Bayesian deep learning is needed in the age of large-scale AI. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39556–39586. PMLR, 2024.
- [1293] George Papandreou and Alan Yuille. Gaussian sampling by local perturbations. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.
- [1294] Eunbyung Park and Junier B Oliva. Meta-curvature. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1295] Trevor Park and George Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.
- [1296] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks, 2013.
- [1297] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1298] Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, Michael O' Boyle, and Amos Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16108–16118. Curran Associates, Inc., 2020.
- [1299] Massimiliano Patacchiola, John Bronskill, Aliaksandra Shysheya, Katja Hofmann, Sebastian Nowozin, and Richard Turner. Contextual squeeze-and-excitation for efficient few-shot image classification. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36680–36692. Curran Associates, Inc., 2022.
- [1300] Dhruvesh Patel, Aishwarya Sahoo, Avinash Amballa, Tahira Naseem, Tim G. J. Rudner, and Andrew McCallum. Insertion language models: Sequence generation with arbitrary-position insertions, 2025.

- [1301] Dhruvesh Patel, Benjamin Rozonoyer, Soumitra Das, Tahira Naseem, Tim G. J. Rudner, and Andrew McCallum. A continuous time Markov chain framework for insertion language models. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.
- [1302] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive Fourier neural operators, 2022.
- [1303] Daniel Paulin, Peter A. Whalley, Neil K. Chada, and Benedict J. Leimkuhler. Sampling from Bayesian neural network posteriors with symmetric minibatch splitting Langevin dynamics. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 5014–5022. PMLR, 2025.
- [1304] Nick Pawlowski, Daniel Coelho de Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 857–869. Curran Associates, Inc., 2020.
- [1305] Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084. PMLR, 2018.
- [1306] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately Bayesian ensembling. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 234–244. PMLR, 2020.
- [1307] Judea Pearl. *Causality*. Cambridge university press, 2000.
- [1308] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 2016. PMLR.
- [1309] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.
- [1310] C. S. Peirce. Note on the theory of the economy of research. *Report of the Superintendent of the United States Coast Survey Showing the Progress of the Work for the Fiscal Year Ending with June, 1876*, 1879.
- [1311] Stefano Peluchetti, Stefano Favaro, and Sandra Fortini. Stable behaviour of infinitely wide deep neural networks. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1137–1146. PMLR, 2020.
- [1312] Alina Peluso, Ioana Danciu, Hong-Jun Yoon, Jamaludin Mohd Yusof, Tanmoy Bhattacharya, Adam Spannaus, Noah Schaefferkoetter, Eric B Durbin, Xiao-Cheng Wu, Antoinette Stroup, et al. Deep learning uncertainty quantification for clinical text classification. *Journal of Biomedical Informatics*, 149:104576, 2024.
- [1313] Feng Peng, Robert A. Jacobs, and Martin A. Tanner. Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *Journal of the American Statistical Association*, 91(434):953–960, 1996.
- [1314] María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22(227):1–40, 2021.

- [1315] Guimar Pescador-Barrios, Sarah Lucie Filippi, and Mark Van Der Wilk. Adjusting model size in continual Gaussian processes: How big is big enough? In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48974–49000. PMLR, 2025.
- [1316] Jonas Peters, Joris Mooij, Dominik Janzing, and Bernhard Schölkopf. Identifiability of causal graphs using functional models, 2012.
- [1317] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [1318] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT press, 2017.
- [1319] Francesca Petralia, Vinayak Rao, and David Dunson. Repulsive mixtures. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [1320] Richard Pettigrew. *Dutch book arguments*. Cambridge University Press, 2020.
- [1321] Henning Petzka, Martin Trimmel, and Cristian Sminchisescu. Notes on the symmetries of 2-layer ReLU-networks. *Proceedings of the Northern Lights Deep Learning Workshop*, 1:6, 2020.
- [1322] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 239–247, New York, NY, USA, 2013. Association for Computing Machinery.
- [1323] Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in numpyro, 2019.
- [1324] Angus Phillips, Gavin Kerrigan, and Tom Rainforth. Bayesian experimental design via score matching, 2026.
- [1325] Juho Piironen and Aki Vehtari. On the Hyperprior Choice for the Global Shrinkage Parameter in the Horseshoe Prior. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 905–913. PMLR, 2017.
- [1326] François Pimont, Hélène Fargeon, Thomas Opitz, Julien Ruffault, Renaud Barbero, Nicolas Martin-StPaul, Eric Rigolot, Miguel Riviere, and Jean-Luc Dupuy. Prediction of regional wildfire activity in the probabilistic Bayesian framework of firelihood. *Ecological applications*, 31(5):e02316, 2021.
- [1327] Dan Piponi, Dave Moore, and Joshua V. Dillon. Joint distributions for TensorFlow probability, 2020.
- [1328] Fabrizio Pittorino, Antonio Ferraro, Gabriele Perugini, Christoph Feinauer, Carlo Baldassi, and Riccardo Zecchina. Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17759–17781. PMLR, 2022.
- [1329] Gábor Pituk, Vik Shirvaikar, and Tom Rainforth. Do Bayesian neural networks actually behave like Bayesian models? In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 49420–49458. PMLR, 2025.
- [1330] Geoff Pleiss and John Cunningham. The limitations of large width in neural networks: A deep Gaussian process perspective. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3349–3363. Curran Associates, Inc., 2021.

- [1331] Martyn Plummer. Cuts in Bayesian graphical models. *Statistics and Computing*, 25(1):37–43, 2015.
- [1332] ShaoWei Png and Joelle Pineau. Bayesian reinforcement learning for pomdp-based dialogue systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 2156–2159. IEEE, 2011.
- [1333] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 1862–1874, 2024.
- [1334] Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1335] Nicholas G Polson and Veronika Ročková. Posterior concentration for sparse deep learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1336] Nicholas G. Polson and James G. Scott. On the Half-Cauchy Prior for a Global Scale Parameter. *Bayesian Analysis*, 7(4):887 – 902, 2012.
- [1337] Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using pólya-gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349, 2013.
- [1338] Boris Polyak. New stochastic approximation type procedures. *Avtomatica i Telemekhanika*, 7:98–107, 1990.
- [1339] Yury Polyanskiy and Yihong Wu. Strong data-processing inequalities for channels and bayesian networks. In Eric Carlen, Mokshay Madiman, and Elisabeth M. Werner, editors, *Convexity and Concentration*, pages 211–249, New York, NY, 2017. Springer New York.
- [1340] Yury Polyanskiy and Yihong Wu. *Information theory: From coding to learning*. Cambridge university press, 2025.
- [1341] Pedro Ponte and Roger G. Melko. Kernel methods for interpretable machine learning of order parameters. *Phys. Rev. B*, 96:205146, 2017.
- [1342] Arya A. Pourzanjani, Richard M. Jiang, and Linda R. Petzold. Improving the identifiability of neural networks for Bayesian inference. In *Advances in Neural Information Processing Systems: Deep Learning Workshop*, 2017.
- [1343] Ameya Prabhu, Charles Dognin, and Maneesh Singh. Sampling bias in deep active classification: An empirical study. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4058–4068, Hong Kong, China, 2019. Association for Computational Linguistics.
- [1344] Dennis Prangle, Sophie Harbisher, and Colin S. Gillespie. Bayesian Experimental Design Without Posterior Calculations: An Adversarial Approach. *Bayesian Analysis*, 18(1):133 – 163, 2023.
- [1345] R. Price. A useful theorem for nonlinear devices having Gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.
- [1346] Victor Priser, Pascal Bianchi, and Adil Salim. Long-time asymptotics of noisy SVGD outside the population limit, 2024.
- [1347] Friedrich Pukelsheim. *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, 2006.
- [1348] Gintaras V. Puskorius and Lee A. Feldkamp. *Parameter-Based Kalman Filter Training: Theory and Implementation*, chapter 2, pages 23–67. John Wiley & Sons, Ltd, 2001.

- [1349] G.V. Puskorius and L.A. Feldkamp. Decoupled extended Kalman filter training of feedforward layered networks. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume i, pages 771–777 vol.1, 1991.
- [1350] Patrick Pynadath, Riddhiman Bhattacharya, Arun Hariharan, and Ruqi Zhang. Gradient-based discrete sampling with automatic cyclical scheduling. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 46728–46763. Curran Associates, Inc., 2024.
- [1351] Yichen Qian, Ming Lin, Xiuyu Sun, Zhiyu Tan, and Rong Jin. Entroformer: A transformer-based entropy model for learned image compression, 2022.
- [1352] Yeping Lina Qiu, Hong Zheng, and Olivier Gevaert. Genomic data imputation with variational auto-encoders. *GigaScience*, 9(8):giaa082, 2020.
- [1353] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 50817–50847. PMLR, 2025.
- [1354] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.
- [1355] Ross Quinlan. Statlog (Australian Credit Approval). UCI Machine Learning Repository, 1987. DOI: <https://doi.org/10.24432/C59012>.
- [1356] Fernando A. Quintana, Peter Müller, Alejandro Jara, and Steven N. MacEachern. The Dependent Dirichlet Process and Related Models. *Statistical Science*, 37(1):24 – 41, 2022.
- [1357] Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019.
- [1358] Alberto Racca and Luca Magri. Robust optimization and validation of echo state networks for learning chaotic dynamics. *Neural Networks*, 142:252–268, 2021.
- [1359] Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1452–1466, 2022.
- [1360] Stefan T. Radev, Marvin Schmitt, Valentin Pratz, Umberto Picchini, Ullrich Köthe, and Paul-Christian Bürkner. Jana: Jointly amortized neural approximation of complex Bayesian models, 2023.
- [1361] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
- [1362] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1674–1703. PMLR, 2017.
- [1363] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- [1364] Raiffa and Schlaifer. *Applied Statistical Decision Theory*. Boston: Division of Research, Graduate School of Business Administration, Harvard University, 1961.
- [1365] Tom Rainforth. *Automating inference, learning, and design using probabilistic programming*. PhD thesis, University of Oxford, 2017.

- [1366] Tom Rainforth. Nesting probabilistic programs. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 249–258. AUAI Press, 2018.
- [1367] Tom Rainforth, Rob Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting Monte Carlo estimators. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4267–4276. PMLR, 2018.
- [1368] Tom Rainforth, Adam Kosior, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4277–4285. PMLR, 2018.
- [1369] Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *Statistical Science*, 39(1):100 – 114, 2024.
- [1370] Tom Rainforth, Freddie Bickford Smith, and Adam Foster. Uncertainty, experimental design and active learning from first principles. *In preparation*, 2026.
- [1371] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [1372] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1373] Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005. Neural Networks and Kernel Methods for Structured Domains.
- [1374] Alexandre Rame, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Leon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28656–28679. PMLR, 2023.
- [1375] Poornima Ramesh, Jan-Matthis Lueckmann, Jan Boelts, Álvaro Tejero-Cantero, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. Gatsbi: Generative adversarial training for simulation-based inference, 2022.
- [1376] Ramsey. Truth and probability. *Studies in Subjective Probability*, 1926.
- [1377] Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland, 2014. PMLR.
- [1378] Rajesh Ranganath, Linpeng Tang, Laurent Charlin, and David M. Blei. Deep exponential families, 2014.
- [1379] C. Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 1945.
- [1380] Carl Rasmussen and Zoubin Ghahramani. Occam's razor. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.
- [1381] Carl Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.
- [1382] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

- [1383] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing.
- [1384] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR, 2017.
- [1385] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2912–2921. PMLR, 2017.
- [1386] Brian J Reich and Benjamin A Shaby. A hierarchical max-stable spatial model for extreme precipitation. *The annals of applied statistics*, 6(4):1430, 2012.
- [1387] Isaac Reid, Krzysztof Marcin Choromanski, Valerii Likhoshesterov, and Adrian Weller. Simplex random features. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28864–28888. PMLR, 2023.
- [1388] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.
- [1389] Matthew Reimherr, Xiao-Li Meng, and Dan L. Nicolae. Prior sample size extensions for assessing prior impact and prior-likelihood discordance. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 83(3):413–437, 2021.
- [1390] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 2015. PMLR.
- [1391] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Beijing, China, 2014. PMLR.
- [1392] Benjamin Rhodes, Kai Xu, and Michael Gutmann. Telescoping density-ratio estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4905–4916. Curran Associates, Inc., 2020.
- [1393] Marina Riabiz, Wilson Ye Chen, Jon Cockayne, Pawel Swietach, Steven A. Niederer, Lester Mackey, and Chris. J. Oates. Optimal thinning of MCMC output. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(4):1059–1081, 2022.
- [1394] G.T. Ribeiro, J.G. Sauer, N. Fracchanabbia, V.C. Mariani, and L.D.S. Coelho. Bayesian optimized echo state network applied to short-term load forecasting. *Energies*, 13:2390, 2020. <https://doi.org/10.3390/en13092390>.
- [1395] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.
- [1396] Giuseppe Riccardi and Dilek Hakkani-Tür. Active and unsupervised learning for automatic speech recognition. In *8th European Conference on Speech Communication and Technology, EUROSPEECH 2003 - INTERSPEECH 2003, Geneva, Switzerland, September 1-4, 2003*, pages 1825–1828. ISCA, 2003.
- [1397] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for thompson sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

- [1398] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8583–8595. Curran Associates, Inc., 2021.
- [1399] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- [1400] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162, 1979.
- [1401] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured Laplace approximations for overcoming catastrophic forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1402] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [1403] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527 – 535, 1952.
- [1404] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.
- [1405] Christian P. Robert. *The Bayesian Choice*. Springer, 2007.
- [1406] Christian P. Robert and George Casella. *Monte Carlo statistical methods*. Springer, 1999.
- [1407] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, 2022.
- [1408] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110 – 120, 1997.
- [1409] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.
- [1410] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.
- [1411] Gareth O Roberts, Jeffrey S Rosenthal, and Peter O Schwartz. Convergence properties of perturbed Markov chains. *Journal of applied probability*, 35(1):1–11, 1998.
- [1412] Harry V. Roberts. Probabilistic prediction. *Journal of the American Statistical Association*, 60(309):50–62, 1965.
- [1413] David S. Robertson, Kim May Lee, Boryana C. López-Kolkovska, and Sofía S. Villar. Response-Adaptive Randomization in Clinical Trials: From Myths to Practical Considerations. *Statistical Science*, 38(2):185 – 208, 2023.
- [1414] Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-pfn: In-context learning for causal effect estimation, 2025.
- [1415] Jakob Robnik, G. Bruno De Luca, Eva Silverstein, and Uroš Seljak. Microcanonical Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 24(311):1–34, 2023.
- [1416] Karl Rohe and Muzhe Zeng. Vintage factor analysis with varimax performs statistical inference. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(4):1037–1060, 2023.
- [1417] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

- [1418] David W. Romero and Suhas Lohit. Learning partial equivariances from data. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36466–36478. Curran Associates, Inc., 2022.
- [1419] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [1420] Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with gears. *Nature Biotechnology*, 42(6):927–935, 2024.
- [1421] Jeffrey S Rosenthal. Parallel computing and Monte Carlo algorithms. *Far east journal of theoretical statistics*, 4(2):207–236, 2000.
- [1422] Breeshey Roskams-Hieter, Jude Wells, and Sara Wade. Leveraging variational autoencoders for multiple data imputation. In *Machine Learning and Knowledge Discovery in Databases: Research Track: European Conference, ECML PKDD 2023, Turin, Italy, September 18–22, 2023, Proceedings, Part I*, pages 491–506, Berlin, Heidelberg, 2023. Springer-Verlag.
- [1423] Stéphane Ross, Joelle Pineau, Brahim Chaib-draa, and Pierre Kreitmann. A Bayesian approach for learning and planning in partially observable Markov decision processes. *Journal of Machine Learning Research*, 12(48):1729–1770, 2011.
- [1424] Tom Rossa, Angus Phillips, and Tom Rainforth. Action-BED: Task-driven Bayesian experimental design with singly intractable objectives, 2026.
- [1425] Simone Rossi, Markus Heinonen, Edwin Bonilla, Zheyang Shen, and Maurizio Filippone. Sparse Gaussian processes revisited: Bayesian approaches to inducing-variable approximations. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1837–1845. PMLR, 2021.
- [1426] Nina Röckler, Tassilo Jungnitz, Albrecht Sigler, Alexander Bird, Martin Mittag, Jeong Seop Rhee, Thomas Deller, Hermann Cuntz, Nils Brose, Stephan W. Schwarzacher, and Peter Jedlicka. Skewed distribution of spines is independent of presynaptic transmitter release and synaptic plasticity, and emerges early during adult neurogenesis. *Open Biology*, 13(8):230063, 2023.
- [1427] Hrittik Roy, Marco Miani, Carl Henrik Ek, Philipp Hennig, Marvin Pförtner, Lukas Tatzel, and Søren Hauberg. Reparameterization invariance in approximate Bayesian inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 8132–8164. Curran Associates, Inc., 2024.
- [1428] Jean-François Roy, François Laviolette, and Mario Marchand. From PAC-Bayes bounds to quadratic programs for majority votes. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 649–656. Omnipress, 2011.
- [1429] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In Carla E. Brodley and Andrea Pohorecký Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 441–448. Morgan Kaufmann, 2001.
- [1430] François Rozet and Gilles Louppe. Arbitrary marginal neural ratio estimation for simulation-based inference, 2021.
- [1431] Ruberg, Beckers, Hemmings, Honig, Irony, LaVange, Lieberman, Mayne, and Moscicki. Application of Bayesian approaches in drug development: starting a virtuous cycle. *Nature Reviews Drug Discovery*, 2023.
- [1432] Donald B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.
- [1433] W. Rudin. Fourier analysis on groups. *Wiley Classics Library. Wiley-Interscience New York, reprint edition*, 1994.

- [1434] Tim G. J. Rudner, Dino Sejdinovic, and Yarin Gal. Inter-domain deep Gaussian processes. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8286–8294. PMLR, 2020.
- [1435] Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractable function-space variational inference in Bayesian neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22686–22698. Curran Associates, Inc., 2022.
- [1436] Tim G. J. Rudner, Sanyam Kapoor, Shikai Qiu, and Andrew Gordon Wilson. Function-space regularization in neural networks: A probabilistic perspective. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 29275–29290. PMLR, 2023.
- [1437] Tim G. J. Rudner, Ya Shi Zhang, Andrew Gordon Wilson, and Julia Kempe. Mind the GAP: Improving robustness to subpopulation shifts with group-aware priors. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 127–135. PMLR, 2024.
- [1438] Tim G. J. Rudner, Xiang Pan, Yucen Lily Li, Ravid Shwartz-Ziv, and Andrew Gordon Wilson. Fine-tuning with uncertainty-aware priors makes vision and language foundation models more reliable. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 154–162. PMLR, 2025.
- [1439] David Rügamer. On the Epistemic Uncertainty of Overparametrized Neural Networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2026.
- [1440] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.
- [1441] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [1442] Massimiliano Russo, Burton H Singer, and David B Dunson. Multivariate mixed membership modeling: inferring domain-specific risk profiles. *The Annals of Applied Statistics*, 16(1):391, 2022.
- [1443] Elizabeth G. Ryan, Christopher C. Drovandi, James M. McGree, and Anthony N. Pettitt. A review of modern computational algorithms for Bayesian optimal design. *International Statistical Review*, 84(1): 128–154, 2016.
- [1444] Kenneth J Ryan. Estimating expected information gains for experimental designs with application to the random fatigue-limit model. *Journal of Computational and Graphical Statistics*, 12(3):585–603, 2003.
- [1445] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R. Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training, 2018.
- [1446] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 130136–130184. Curran Associates, Inc., 2024.
- [1447] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 52584–52619. PMLR, 2025.
- [1448] Teemu Säilynoja, Marvin Schmitt, Paul-Christian Bürkner, and Aki Vehtari. Posterior sbc: Simulation-based calibration checking conditional on data, 2025.

- [1449] Charbel Sakr, Naigang Wang, Chia-Yu Chen, Jungwook Choi, Ankur Agrawal, Naresh Shanbhag, and Kailash Gopalakrishnan. Accumulation bit-width scaling for ultra-low precision training of deep networks, 2019.
- [1450] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, pages 872–879, New York, NY, USA, 2008. Association for Computing Machinery.
- [1451] Adil Salim, Lukang Sun, and Peter Richtarik. A convergence theory for SVGD in the population limit under talagrand’s inequality t1. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19139–19152. PMLR, 2022.
- [1452] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017.
- [1453] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1454] Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. Orthogonally decoupled variational Gaussian processes. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1455] Paul-Marie Samson. Concentration of measure inequalities for Markov chains and Φ -mixing processes. *The Annals of Probability*, 28(1):416 – 461, 2000.
- [1456] Ashwin Samudre, Mircea Petrache, Brian D. Nord, and Shubhendu Trivedi. Symmetry-based structured matrices for efficient approximately equivariant networks, 2024.
- [1457] Sophia Sanborn, Christian Shewmake, Bruno Olshausen, and Christopher Hillar. Bispectral neural networks, 2022.
- [1458] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Springer, 2015.
- [1459] Adhithya Saravanan, Rik Knowles, Gavin Kerrigan, and Tom Rainforth. Diffbed: Scaling Bayesian experimental design to high-dimensions. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 131048–131073, 2026.
- [1460] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*. Cambridge University Press, 2019.
- [1461] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, second edition edition, 2023.
- [1462] Warren S. Sarle. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pages 352–360, 1995.
- [1463] Savage. *The Foundations of Statistics*. John Wiley and Sons, 1954.
- [1464] Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 951–958. Omnipress, 2010.
- [1465] Khalid Sayood. Vector quantization. In *Introduction to Data Compression (Fourth Edition)*, The Morgan Kaufmann Series in Multimedia Information and Systems, pages 295–344. Morgan Kaufmann, Boston, fourth edition edition, 2012.
- [1466] Meyer Scetbon, Joel Jennings, Agrin Hilmkil, Cheng Zhang, and Chao Ma. A fixed-point approach for causal generative modeling, 2024.
- [1467] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In Frank Hoffmann, David J. Hand, Niall Adams, Douglas Fisher, and Gabriela Guimaraes, editors, *Advances in Intelligent Data Analysis*, pages 309–318, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

- [1468] Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.
- [1469] Dominik Scheuer, Frederic Runge, Joerg KH Franke, Michael T Wolfinger, Christoph Flamm, and Frank Hutter. Kinpfm: Bayesian approximation of RNA folding kinetics using prior-data fitted networks. *bioRxiv*, pages 2024–10, 2024.
- [1470] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, pages 222–227, Cambridge, MA, USA, 1991. MIT Press.
- [1471] Marvin Schmitt, Valentin Pratz, Ullrich Köthe, Paul-Christian Bürkner, and Stefan T. Radev. Consistency models for scalable and fast simulation-based inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 126908–126945. Curran Associates, Inc., 2024.
- [1472] Bernhard Schölkopf and Alex J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002.
- [1473] Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.
- [1474] Christopher Schröder, Lydia Müller, Andreas Niekler, and Martin Potthast. Small-text: Active learning for text classification in python. In *Proceedings of the 17th conference of the European chapter of the association for computational linguistics: system demonstrations*, pages 84–95, 2023.
- [1475] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25278–25294. Curran Associates, Inc., 2022.
- [1476] Loraine Schwartz. On consistency of Bayes procedures. *Proceedings of the National Academy of Sciences*, 52(1):46–49, 1964.
- [1477] Lorraine Schwartz. On Bayes procedures. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 4(1):10–26, 1965.
- [1478] Karl Schwarzschild. Über das gravitationsfeld eines massenpunktes nach der einsteinschen theorie. *Sitzungsberichte der königlich preussischen Akademie der Wissenschaften*, pages 189–196, 1916.
- [1479] Tobias Schwedes and Ben Calderhead. Rao-blackwellised parallel MCMC. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3448–3456. PMLR, 2021.
- [1480] Pola Schwöbel, Martin Jørgensen, Sebastian W. Ober, and Mark Van Der Wilk. Last layer marginal likelihood for invariance learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3542–3555. PMLR, 2022.
- [1481] Paul D. Scott and Shaul Markovitch. Learning novel domains through curiosity and conjecture. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI’89*, pages 669–674, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [1482] Othmane Sebbouh, Robert M Gower, and Aaron Defazio. Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 3935–3971. PMLR, 2021.
- [1483] Matthias Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

- [1484] Dinesh Seemakurty, Michael Andrew Maylahn, Mark William Davis, Roheet Bantval Rao, Aditya Patel, Izzatbir Singh, and Aashay Sachdeva. Systems and methods using ensemble machine learning techniques for future event detection, 2021. US Patent 11,195,616.
- [1485] Yevgeny Seldin. Machine learning. the science of selection under uncertainty, 2025.
- [1486] Yevgeny Seldin and Naftali Tishby. PAC-Bayesian analysis of co-clustering and beyond. *Journal of Machine Learning Research*, 11(117):3595–3646, 2010.
- [1487] Yevgeny Seldin, François Laviolette, Nicolò Cesa-Bianchi, John Shawe-Taylor, and Peter Auer. PAC-Bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12):7086–7093, 2012.
- [1488] Torben Sell and Sumeetpal Sidhu Singh. Trace-class Gaussian priors for Bayesian learning of neural networks with MCMC. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(1):46–66, 2023.
- [1489] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [1490] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with sub-word units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics.
- [1491] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1492] Settles. Active learning literature survey. Technical report, University of Wisconsin, Madison, 2009.
- [1493] Settles and Craven. An analysis of active learning for sequence labeling. *Conference on Empirical Methods in Natural Language Processing*, 2008.
- [1494] Burr Settles. *Active Learning*. Springer, 2012.
- [1495] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- [1496] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, pages 287–294, New York, NY, USA, 1992. Association for Computing Machinery.
- [1497] Ben Shababo, Brooks Paige, Ari Pakman, and Liam Paninski. Bayesian inference and online experimental design for mapping neural microcircuits. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [1498] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008.
- [1499] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948.
- [1500] Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *Institute of Radio Engineers, International Convention Record*, 7(4):142–163, 1959.
- [1501] Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do Bayesian neural networks need to be fully stochastic? In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 7694–7722. PMLR, 2023.
- [1502] Mrinank Sharma, Tom Rainforth, Yee Whye Teh, and Vincent Fortuin. Incorporating unlabelled data into Bayesian neural networks. *Transactions on Machine Learning Research*, 2024. Expert Certification.

- [1503] Louis Sharrock and Christopher Nemeth. Coin sampling: Gradient-based Bayesian inference without learning rates. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 30850–30882. PMLR, 2023.
- [1504] Louis Sharrock and Christopher Nemeth. Tuning-free sampling via optimization on the space of probability measures, 2025.
- [1505] Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models, 2022.
- [1506] Louis Sharrock, Lester Mackey, and Christopher Nemeth. Learning rate free sampling in constrained domains. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 65380–65415. Curran Associates, Inc., 2023.
- [1507] John Shawe-Taylor and Robert C. Williamson. A PAC analysis of a Bayesian estimator. In *Proceedings of the Conference on Learning Theory (COLT)*, 1997.
- [1508] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [1509] Xiaotong Shen, Wei Pan, and Yunzhang Zhu. Likelihood-based selection and sharp parameter estimation. *Journal of the American Statistical Association*, 107(497):223–232, 2012. PMID: 22736876.
- [1510] Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning is effective for large deep networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 44665–44686. PMLR, 2024.
- [1511] Chunyang Sheng, Jun Zhao, Wei Wang, and Henry Leung. Prediction intervals for a noisy nonlinear time series based on a bootstrapping reservoir computing network ensemble. *IEEE Transactions on Neural Networks and Learning Systems*, 24(7):1036–1048, 2013.
- [1512] Jiaxin Shi and Lester Mackey. A finite-particle convergence rate for Stein variational gradient descent. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 26831–26844. Curran Associates, Inc., 2023.
- [1513] Jiaxin Shi, Michalis Titsias, and Andriy Mnih. Sparse orthogonal variational inference for Gaussian processes. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1932–1942. PMLR, 2020.
- [1514] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 103131–103167. Curran Associates, Inc., 2024.
- [1515] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.
- [1516] Daniel Simpson, Håvard Rue, Andrea Riebler, Thiago G. Martins, and Sigrunn H. Sørbye. Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. *Statistical Science*, 32(1):1 – 28, 2017.
- [1517] Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference*

- on *Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9722–9732. PMLR, 2021.
- [1518] Sharad Singhal and Lance Wu. Training multilayer perceptrons with the extended Kalman algorithm. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.
 - [1519] Sandip Sinharay, Hal S Stern, and Daniel Russell. The use of multiple imputation for the analysis of missing data. *Psychological methods*, 6(4):317, 2001.
 - [1520] S. A. Sisson, Y. Fan, and M. A. Beaumont. Overview of approximate Bayesian computation, 2018.
 - [1521] Lars Skaaret-Lund, Geir Storvik, and Aliaksandr Hubin. Sparsifying Bayesian neural networks with latent binary variables and normalizing flows. *Transactions on Machine Learning Research*, 2024.
 - [1522] Joanna Sliwa, Frank Schneider, Nathanael Bosch, Agustinus Kristiadi, and Philipp Hennig. Low-rank filtering and smoothing for sequential deep learning, 2024.
 - [1523] Marek Smieja and Jacek Tabor. Entropy of the mixture of sources and entropy dimension. *IEEE Transactions on Information Theory*, 58(5):2719–2728, 2012.
 - [1524] Gerald L Smith, Stanley F Schmidt, and Leonard A McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. Technical Report NASA-TR-R-135, NASA, 1962.
 - [1525] Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1-2):1–85, 1918.
 - [1526] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection, 2018.
 - [1527] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
 - [1528] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.
 - [1529] Edward Snelson, Zoubin Ghahramani, and Carl Rasmussen. Warped Gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.
 - [1530] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 2015. PMLR.
 - [1531] Elliott Sober. *Ockham’s Razors: A User’s Manual*. Cambridge University Press, 2015.
 - [1532] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 2015. PMLR.
 - [1533] Emanuel Sommer and David Rügamer. Position: The time for sampling is now! charting a new course for Bayesian deep learning. In *Forty-third International Conference on Machine Learning Position Paper Track*, 2026.

- [1534] Emanuel Sommer, Lisa Wimmer, Theodore Papamarkou, Ludwig Bothmann, Bernd Bischl, and David Rügamer. Connecting the dots: Is mode-connectedness the key to feasible sample-based inference in Bayesian neural networks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 45988–46018. PMLR, 2024.
- [1535] Emanuel Sommer, Jakob Robnik, Giorgi Nozadze, Uros Seljak, and David Rügamer. Microcanonical Langevin ensembles: Advancing the sampling of Bayesian neural networks. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 15101–15122, 2025.
- [1536] Emanuel Sommer, Kangning Diao, Jakob Robnik, Uroš Seljak, and David Rügamer. Can Microcanonical Langevin Dynamics Leverage Mini-Batch Gradient Noise? In *International Conference on Machine Learning (ICML)*. PMLR, 2026.
- [1537] Sonehara, Kawato, Miyake, and Nakane. Image data compression using a neural network model. In *International 1989 Joint Conference on Neural Networks*, pages 35–41 vol.2, 1989.
- [1538] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.
- [1539] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc., 2020.
- [1540] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1541] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2020.
- [1542] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 2023.
- [1543] Zhouhui Song, Zhenyu Liu, and Dongsheng Wang. Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 816–823. AAAI Press, 2018.
- [1544] Ankit Sonthalia, Alexander Rubinstein, Ehsan Abbasnejad, and Seong Joon Oh. Do deep neural network solutions form a star domain? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- [1545] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [1546] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, second edition edition, 2001.
- [1547] V Spokoiny and M Panov. Accuracy of Gaussian approximation for highdimensional posterior distributions. *Bernoulli*, 2025.
- [1548] M. D. Springer and W. E. Thompson. The distribution of products of beta, gamma and Gaussian random variables. *SIAM Journal on Applied Mathematics*, 18(4):721–737, 1970.

- [1549] Sharang M. Sriramu, Rochelle Barsz, Elizabeth Polito, and Aaron B. Wagner. Fast channel simulation via error-correcting codes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 107932–107959. Curran Associates, Inc., 2024.
- [1550] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. Featured Certification.
- [1551] M.S. Srivastava. Singular Wishart and multivariate beta distributions. *The Annals of Statistics*, 31(5):1537 – 1560, 2003.
- [1552] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014.
- [1553] Joe Staines and David Barber. Variational optimization, 2012.
- [1554] Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.
- [1555] Megan Stanley, John F Bronskill, Krzysztof Maziarsz, Hubert Misztela, Jessica Lanini, Marwin Segler, Nadine Schneider, and Marc Brockschmidt. FS-Mol: A few-shot learning dataset of molecules. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
- [1556] Ronald Fedkiw Stanley Osher. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
- [1557] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20459–20478. PMLR, 2022.
- [1558] Vladimir Starostin, Maximilian Dax, Alexander Gerlach, Alexander Hinderhofer, Álvaro Tejero-Cantero, and Frank Schreiber. Fast and reliable probabilistic reflectometry inversion with prior-amortized neural posterior estimation. *Science Advances*, 11(11):eadr9668, 2025.
- [1559] Charles Stein. Approximate computation of expectations. In *Institute of Mathematical Statistics Lecture Notes*. IMS, 1986.
- [1560] Charles M. Stein. Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9(6):1135 – 1151, 1981.
- [1561] Michael Steiner. Towards quantifying non-local information transfer: finite-bit non-locality. *Physics Letters A*, 270(5):239–244, 2000.
- [1562] Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.
- [1563] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.
- [1564] Pierre Stock, Benjamin Graham, Rémi Gribonval, and Hervé Jégou. Equi-normalization of neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [1565] B. J. Stocks, B. D. Lawson, M. E. Alexander, C. E. Van Wagner, R. S. McAlpine, T. J. Lynham, and D. E. Dubé. The canadian forest fire danger rating system: An overview. *The Forestry Chronicle*, 65(6):450–457, 1989.

- [1566] George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 29215–29237, 2024.
- [1567] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.
- [1568] Michael A Storey, Michael Bedward, Owen F Price, Ross A Bradstock, and Jason J Sharples. Derivation of a Bayesian fire spread model using large-scale wildfire observations. *Environmental Modelling & Software*, 144:105127, 2021.
- [1569] Malcolm J. A. Strens. A Bayesian framework for reinforcement learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, Stanford University, Stanford, CA, USA, June 29 - July 2, 2000, pages 943–950. Morgan Kaufmann, 2000.
- [1570] A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.
- [1571] Mahesh Subedar, Ranganath Krishnan, Sidharth N Kashyap, and Omesh Tickoo. Quantization of Bayesian neural networks and its effect on quality of uncertainty. In *Workshop on Uncertainty and Robustness in Deep Learning, ICML*, 2021.
- [1572] Masashi Sugiyama. Active learning for misspecified models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.
- [1573] Chenxi Sun, Moxian Song, Derun Cai, Baofeng Zhang, Shenda Hong, and Hongyan Li. A systematic review of echo state networks from design to application. *IEEE Transactions on Artificial Intelligence*, 5(1):23–37, 2024.
- [1574] Haoran Sun, Bo Dai, Charles Sutton, Dale Schuurmans, and Hanjun Dai. Any-scale balanced samplers for discrete space. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [1575] Lukang Sun, Avetik Karagulyan, and Peter Richtarik. Convergence of Stein variational gradient descent under a weaker smoothness condition. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 3693–3717. PMLR, 2023.
- [1576] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger B. Grosse. Functional variational Bayesian neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [1577] Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi (Viji) Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1578] Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi (Viji) Srinivasan, and Kailash Gopalakrishnan. Ultra-low precision 4-bit training of deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1796–1807. Curran Associates, Inc., 2020.
- [1579] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: an Introduction*. MIT Press, second edition edition, 2018.
- [1580] Kevin Swersky, Jasper Snoek, and Ryan Adams. Multi-task Bayesian optimization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

- [1581] Saifuddin Syed, Alexandre Bouchard-Côté, Kevin Chern, and Arnaud Doucet. Optimised annealed sequential Monte Carlo samplers, 2024.
- [1582] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.
- [1583] Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Timepfm: Effective multivariate time series forecasting with synthetic data. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 20761–20769. AAAI Press, 2025.
- [1584] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2018.
- [1585] Sean Talts, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. Validating Bayesian inference algorithms with simulation-based calibration, 2018.
- [1586] Kunal Talwar. Computational separations between sampling and optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1587] Alex Tamkin, Dat Nguyen, Salil Deshpande, Jesse Mu, and Noah Goodman. Active learning helps pre-trained models learn the intended task. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 28140–28153. Curran Associates, Inc., 2022.
- [1588] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.
- [1589] Wei Tan, Lan Du, and Wray Buntine. Diversity enhanced active learning with strictly proper scoring rules. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 10906–10918. Curran Associates, Inc., 2021.
- [1590] Roubing Tang, Sabina J. Sloman, and Samuel Kaski. Representative, informative, and de-amplifying: Requirements for robust Bayesian active learning under model misspecification. In *International Conference on Artificial Intelligence and Statistics*, 2026.
- [1591] Alexander Tarr and Kosuke Imai. Estimating average treatment effects with support vector machines, 2021.
- [1592] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers, 2021.
- [1593] Nazanin Tehrani, Nimar S. Arora, Yucen Lily Li, Kinjal Divesh Shah, David Noursi, Michael Tingley, Narjes Torabi, Sepehr Masouleh, Eric Lippert, and Erik Meijer. Bean machine: A declarative probabilistic programming language for efficient programmable inference. In Manfred Jaeger and Thomas Dyhre Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 485–496. PMLR, 2020.
- [1594] Joshua B. Tenenbaum, Thomas L. Griffiths, and Charles Kemp. Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 2006.
- [1595] The COVID Moonshot Consortium. Open science discovery of oral non-covalent sars-cov-2 main protease inhibitor therapeutics. *bioRxiv*, 2022.
- [1596] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [1597] Lucas Theis, Tim Salimans, Matthew D. Hoffman, and Fabian Mentzer. Lossy compression with Gaussian diffusion, 2022.

- [1598] Christina V. Theodoris, Ling Xiao, Anant Chopra, Mark D. Chaffin, Zeina R. Al Sayed, Matthew C. Hill, Helene Mantineo, Elizabeth M. Brydon, Zexian Zeng, X. Shirley Liu, and Patrick T. Ellinor. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.
- [1599] Niklas Thiemann, Christian Igel, Olivier Wintenberger, and Yevgeny Seldin. A strongly quasiconvex PAC-Bayesian bound. In Steve Hanneke and Lev Reyzin, editors, *Proceedings of the 28th International Conference on Algorithmic Learning Theory*, volume 76 of *Proceedings of Machine Learning Research*, pages 466–492. PMLR, 2017.
- [1600] Anna Thomas, Albert Gu, Tri Dao, Atri Rudra, and Christopher Ré. Learning compressed transforms with low displacement rank. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1601] Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. Likelihood-free inference by ratio estimation, 2016.
- [1602] Philip S Thomas, Bruno Castro da Silva, Andrew G Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. volume 366, pages 999–1004, 2019.
- [1603] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.
- [1604] Sebastian B. Thrun and Knut Möller. Active exploration in dynamic environments. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.
- [1605] Changxin Tian, Kunlong Chen, Jia Liu, Ziqi Liu, Zhiqiang Zhang, and Jun Zhou. Towards greater leverage: Scaling laws for efficient mixture-of-experts language models, 2025.
- [1606] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020.
- [1607] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
- [1608] Luke Tierney. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701 – 1728, 1994.
- [1609] Luke Tierney and Joseph B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.
- [1610] Petru Tighineanu, Kathrin Skubch, Paul Baireuther, Attila Reiss, Felix Berkenkamp, and Julia Vinograd-ska. Transfer learning with Gaussian processes for Bayesian optimization. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 6152–6181. PMLR, 2022.
- [1611] Gleb Tikhonov, Øystein H Opedal, Nerea Abrego, Aleksi Lehtikainen, Melinda MJ de Jonge, Jari Oksanen, and Otso Ovaskainen. Joint species distribution modelling with the R-package HMSC. *Methods in Ecology and Evolution*, 11(3):442–447, 2020.
- [1612] N. Tishby, E. Levin, and S. A. Solla. Consistent inference of probabilities in layered networks: Predictions and generalizations. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 403–409, 1989.
- [1613] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.
- [1614] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000.

- [1615] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.
- [1616] Michalis Titsias and Neil D. Lawrence. Bayesian Gaussian process latent variable model. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 844–851, Chia Laguna Resort, Sardinia, Italy, 2010. PMLR.
- [1617] Michalis Titsias, Alexandre Galashov, Amal Rannen-Triki, Razvan Pascanu, Yee Whye Teh, and Jorg Bornschein. Kalman filter for online classification of non-stationary data. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 34798–34816, 2024.
- [1618] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks, 2015.
- [1619] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.
- [1620] Ilya O Tolstikhin and Yevgeny Seldin. PAC-Bayes-empirical-Bernstein inequality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [1621] Jakub Tomczak and Max Welling. VAE with a vampprior. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 2018.
- [1622] Marcin Tomczak, Siddharth Swaroop, and Richard Turner. Efficient low rank Gaussian variational inference for neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4610–4622. Curran Associates, Inc., 2020.
- [1623] Jean-Francois Ton, Seth Flaxman, Dino Sejdinovic, and Samir Bhatt. Spatial mapping with Gaussian processes and nonstationary Fourier features. *Journal of Spatial Statistics*, 28:59–78, 2018.
- [1624] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning, 2019.
- [1625] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [1626] James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: lossless image compression with hierarchical latent variable models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [1627] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, Pietro Michiardi, Edwin Bonilla, and Maurizio Filippone. Model selection for Bayesian autoencoders. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19730–19742. Curran Associates, Inc., 2021.
- [1628] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for Bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022.
- [1629] Dustin Tran, Mike Dusenberry, Mark van der Wilk, and Danijar Hafner. Bayesian layers: A module for neural network uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

- [1630] Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1631] Volker Tresp. Mixtures of Gaussian processes. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.
- [1632] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. Springer, second edition edition, 2009.
- [1633] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [1634] Kostas Triantafyllopoulos. *Bayesian Inference of State Space Models: Kalman Filtering and Beyond*. Springer, 2021.
- [1635] Brian L Trippe and Richard E Turner. Conditional density estimation with Bayesian normalising flows, 2018.
- [1636] Matthias C.M. Troffaes and Gert de Cooman. *Lower Previsions*. John Wiley & Sons, 2014.
- [1637] Filip Tronarp, Ángel F. García-Fernández, and Simo Särkkä. Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments. *IEEE Signal Processing Letters*, 25(3):408–412, 2018.
- [1638] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.
- [1639] Koji Tsuda, Taishin Kin, and Kiyoshi Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18(suppl_1):S268–S275, 2002.
- [1640] Loring W. Tu. *An Introduction to Manifolds*. Universitext. Springer, New York, NY, 2011.
- [1641] Ruibo Tu, Chao Ma, and Cheng Zhang. Causal-discovery performance of chatgpt in the context of neuropathic pain diagnosis, 2023.
- [1642] Ruibo Tu, Hedvig Kjellström, Gustav Eje Henter, and Cheng Zhang. Carl-gt: Evaluating causal reasoning capabilities of large language models, 2024.
- [1643] Tur, Hakkani-Tür, and Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 2005.
- [1644] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite Markov decision processes with Gaussian processes. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1645] Naonori Ueda and Zoubin Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(10):1223–1241, 2002.
- [1646] Harald Uhlig. On Singular Wishart and Singular Multivariate Beta Distributions. *The Annals of Statistics*, 22(1):395 – 405, 1994.
- [1647] Anders Kirk Uhrenholt, Valentin Charvet, and Bjørn Sand Jensen. Probabilistic selection of inducing points in sparse Gaussian processes. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1035–1044. PMLR, 2021.
- [1648] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016.

- [1649] Ivan Ustyuzhaninov, Ieva Kazlauskaitė, Markus Kaiser, Erik Bodin, Neill Campbell, and Carl Henrik Ek. Compositional uncertainty in deep Gaussian processes. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 480–489. PMLR, 2020.
- [1650] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11287–11302. Curran Associates, Inc., 2021.
- [1651] Valentin, Kleinegesse, Bramley, Series, Gutmann, and Lucas. Designing optimal behavioral experiments using machine learning. *eLife*, 2024.
- [1652] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
- [1653] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9690–9700. PMLR, 2020.
- [1654] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty, 2021.
- [1655] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 2016. PMLR.
- [1656] Dirk van der Hoeven, Tim van Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 2067–2092. PMLR, 2018.
- [1657] Putri A. van der Linden, Alejandro García-Castellanos, Sharvaree Vadgama, Thijs P. Kuipers, and Erik J. Bekkers. Learning symmetries via weight-sharing with doubly stochastic tensors. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 42738–42764. Curran Associates, Inc., 2024.
- [1658] Tycho van der Ouderaa, Alexander Immer, and Mark van der Wilk. Learning layer-wise equivariances automatically using gradients. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 28365–28377. Curran Associates, Inc., 2023.
- [1659] Tycho F. A. van der Ouderaa, David W. Romero, and Mark van der Wilk. Relaxing equivariance constraints with non-stationary continuous filters, 2022.
- [1660] Tycho F. A. van der Ouderaa, Mark van der Wilk, and Pim de Haan. Noether's razor: Learning conserved quantities. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 135943–135965. Curran Associates, Inc., 2024.
- [1661] Tycho F.A. van der Ouderaa and Mark van der Wilk. Learning invariant weights in neural networks. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1992–2001. PMLR, 2022.
- [1662] Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1663] Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes, 2020.

- [1664] D.C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T.E.J. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S.W. Curtiss, S. Della Penna, D. Feinberg, M.F. Glasser, N. Harel, A.C. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S.E. Petersen, F. Prior, B.L. Schlaggar, S.M. Smith, A.Z. Snyder, J. Xu, and E. Yacoub. The human connectome project: A data acquisition perspective. *NeuroImage*, 62(4):2222–2231, 2012. Connectivity.
- [1665] Twan van Laarhoven. L2 regularization versus batch and weight normalization, 2017.
- [1666] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on CPUs. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.
- [1667] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 1982.
- [1668] Roman Vashurin, Maiya Goloburda, Albina Ilina, Aleksandr Rubashevskii, Preslav Nakov, Artem Shelmanov, and Maxim Panov. Uncertainty quantification for LLMs through minimum Bayes risk: Bridging confidence and consistency, 2025.
- [1669] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [1670] Richard D. De Veaux and David J. Hand. How to Lie with Bad Data. *Statistical Science*, 20(3):231 – 238, 2005.
- [1671] A. Vehtari, S. Sarkka, and J. Lampinen. On MCMC sampling in Bayesian mlp neural networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, 2000.
- [1672] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-Normalization, Folding, and Localization: An Improved $\hat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667 – 718, 2021.
- [1673] Gilles Venturini. AGIL: solving the exploration versus exploration dilemma in a single classifier system applied to simulated robotics. In Derek H. Sleeman and Peter Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, UK, July 1-3, 1992*, pages 458–463. Morgan Kaufmann, 1992.
- [1674] Christelle Vergé, Cyrille Dufour, Pierre Del Moral, and Eric Moulines. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing*, 25(2):243–260, 2015.
- [1675] Yogesh Verma, Ayush Bharti, and Vikas Garg. Robust simulation-based inference under missing data via neural processes, 2025.
- [1676] Jeroen K. Vermunt. Latent class modeling with covariates: Two improved three-step approaches. *Political Analysis*, 18(4):450–469, 2017.
- [1677] Julius Vetter, Manuel Gloeckler, Daniel Gedon, and Jakob H. Macke. Effortless, simulation-efficient Bayesian inference using tabular foundation models, 2025.
- [1678] Beate Vieth, Christoph Ziegenhain, Swati Parekh, Wolfgang Enard, and Ines Hellmann. powsimr: power analysis for bulk and single cell RNA-seq experiments. *Bioinformatics*, 33(21):3486–3488, 2017.
- [1679] Cédric Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer, 2009.
- [1680] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.
- [1681] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

- [1682] Mariia Vladimirova, Jakob Verbeek, Pablo Mesejo, and Julyan Arbel. Understanding priors in Bayesian neural networks at the unit level. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6458–6467. PMLR, 2019.
- [1683] Mariia Vladimirova, Stéphane Girard, Hien Nguyen, and Julyan Arbel. Sub-weibull distributions: Generalizing sub-Gaussian and sub-exponential properties to heavier tailed distributions. *Stat*, 9(1):e318, 2020. e318 sta4.318.
- [1684] Mariia Vladimirova, Julyan Arbel, and Stéphane Girard. Bayesian neural network unit priors and generalized Weibull-tail property. In Vineeth N. Balasubramanian and Ivor Tsang, editors, *Proceedings of The 13th Asian Conference on Machine Learning*, volume 157 of *Proceedings of Machine Learning Research*, pages 1397–1412. PMLR, 2021.
- [1685] Mariia Vladimirova, Julyan Arbel, and Stéphane Girard. Dependence between Bayesian neural network units. In *NeurIPS Workshop on Bayesian Deep Learning*, 2021.
- [1686] Mischa von Krause, Stefan T. Radev, and Andreas Voss. Mental speed is high until age 60 as revealed by analysis of over a million participants. *Nature Human Behaviour*, 6(5):700–708, 2022.
- [1687] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.
- [1688] Maxime Vono, Vincent Plassier, Alain Durmus, Aymeric Dieuleveut, and Eric Moulines. Qlsd: Quantised Langevin stochastic dynamics for Bayesian federated learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 6459–6500. PMLR, 2022.
- [1689] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, second edition edition, 2022.
- [1690] C. E. van Wagner. Development and structure of the canadian forest fire weather index system. *Forestry Technical Report - Canadian Forestry Service*, pages viii + 37 pp., 1987.
- [1691] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.
- [1692] Abraham Wald. On the Efficient Design of Statistical Investigations. *The Annals of Mathematical Statistics*, 14(2):134 – 140, 1943.
- [1693] Abraham Wald. An Essentially Complete Class of Admissible Decision Functions. *The Annals of Mathematical Statistics*, 18(4):549 – 555, 1947.
- [1694] Abraham Wald. *Statistical Decision Functions*. Wiley, 1950.
- [1695] A. M. Walker. On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 31(1):80–88, 1969.
- [1696] Stephen Walker. New approaches to Bayesian consistency. *The Annals of Statistics*, 32(5):2028 – 2043, 2004.
- [1697] Stephen G Walker. Bayesian information in an experiment and the Fisher information distance. *Statistics and Probability Letters*, 2016.
- [1698] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.
- [1699] Peter Walley. *Statistical Reasoning with Imprecise Probabilities*. Monographs on Statistics and Applied Probability. Chapman and Hall, 1991.

- [1700] Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Øyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM’s version). In *Second Conference on Language Modeling*, 2025.
- [1701] Benjie Wang, Joel Jennings, and Wenbo Gong. Neural structure learning with stochastic differential equations, 2023.
- [1702] Chaoqi Wang, Shengyang Sun, and Roger Grosse. Beyond marginal uncertainty: How accurately can Bayesian regression models estimate posterior predictive correlations? In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2476–2484. PMLR, 2021.
- [1703] Chenyang Wang and Yun Yang. Estimating the number of components in finite mixture models via variational approximation, 2024.
- [1704] Dilin Wang, Ziyang Tang, Chandrajit Bajaj, and Qiang Liu. Stein variational gradient descent with matrix-valued kernels. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1705] George Wang, Jesse Hoogland, Stan van Wingerden, Zach Furman, and Daniel Murfet. Differentiation and specialization of attention heads via the refined local learning coefficient. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 101037–101082, 2025.
- [1706] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024.
- [1707] Ming Chen Wang and G. E. Uhlenbeck. On the theory of the brownian motion ii. *Rev. Mod. Phys.*, 17: 323–342, 1945.
- [1708] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1709] Qi Wang, Paul A. Parker, and Robert B. Lund. Hierarchical count echo state network models with application to graduate Student enrollments, 2025.
- [1710] Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23078–23091. PMLR, 2022.
- [1711] Xi Wang, Laurence Aitchison, and Maja Rudolph. LoRA ensembles for large language model fine-tuning, 2023.
- [1712] Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. Property-aware relation networks for few-shot molecular property prediction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17441–17454. Curran Associates, Inc., 2021.
- [1713] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

- [1714] Zheng Wang, Wei Xing, Robert Kirby, and Shandian Zhe. Physics informed deep kernel learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 1206–1218. PMLR, 2022.
- [1715] Zhenhua Wang, Scott H. Holan, and Christopher K. Wikle. Echo state networks for spatio-temporal area-level data, 2024.
- [1716] Zhigang Wang, Yu-Rong Zeng, Sirui Wang, and Lin Wang. Optimizing echo state network with backtracking search optimization algorithm for time series forecasting. *Engineering Applications of Artificial Intelligence*, 81:117–132, 2019.
- [1717] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.
- [1718] Zhiming Wang, Jingjing Dong, and Junpeng Zhang. Multi-model ensemble deep learning method to diagnose covid-19 using chest computed tomography images. *Journal of Shanghai Jiaotong University (Science)*, 27(1):70–80, 2022.
- [1719] Ziyi Wang, Yujie Chen, Qifan Song, and Ruqi Zhang. Enhancing low-precision sampling via stochastic gradient Hamiltonian Monte Carlo. *Transactions on Machine Learning Research*, 2024.
- [1720] Daniel Ward, Patrick Cannon, Mark Beaumont, Matteo Fasiolo, and Sebastian Schmon. Robust neural posterior estimation and statistical model criticism. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 33845–33859. Curran Associates, Inc., 2022.
- [1721] David I. Warton, F. Guillaume Blanchet, Robert B. O’Hara, Otso Ovaskainen, Sara Taskinen, Steven C. Walker, and Francis K.C. Hui. So many variables: Joint modeling in community ecology. *Trends in Ecology & Evolution*, 30(12):766–779, 2015.
- [1722] Sumio Watanabe. Review and prospect of algebraic research in equivalent framework between statistical mechanics and machine learning theory. *Reviews in Mathematical Physics*, 0(0):2461009, 0.
- [1723] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009.
- [1724] Sumio Watanabe. Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11(116):3571–3594, 2010.
- [1725] Sumio Watanabe. A widely applicable Bayesian information criterion. *Journal of Machine Learning Research*, 14(27):867–897, 2013.
- [1726] Sumio Watanabe. *Mathematical Theory of Bayesian Statistics*. CRC Press, 2018.
- [1727] Sumio Watanabe. Recent advances in algebraic geometry and Bayesian statistics. *Information Geometry*, 7(1):187–209, 2024.
- [1728] Steve Waterhouse, David MacKay, and Anthony Robinson. Bayesian methods for mixtures of experts. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.
- [1729] Joe Watson, Jihao Andreas Lin, Pascal Klink, Joni Pajarinen, and Jan Peters. Latent derivative Bayesian last layer networks. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1198–1206. PMLR, 2021.
- [1730] Tobias Weber, Bálint Mucsányi, Lenard Rommel, Thomas Christie, Lars Kasüschke, Marvin Pförtner, and Philipp Hennig. laplax - Laplace approximations with JAX. In *Championing Open-source DEvelopment in ML Workshop ICML25*, 2025.

- [1731] Antoine Wehenkel, Juan L. Gamella, Ozan Sener, Jens Behrmann, Guillermo Sapiro, Jörn-Henrik Jacobsen, and Marco Cuturi. Addressing misspecification in simulation-based inference through data-driven calibration, 2024.
- [1732] Susan Wei and Edmund Lau. Variational Bayesian neural networks via resolution of singularities. *Journal of Computational and Graphical Statistics*, 33(4):1464–1474, 2024.
- [1733] Susan Wei, Daniel Murfet, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that’s good. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10473–10486, 2023.
- [1734] Yufan Wei, Kun Zhou, Lingjun Mao, Zijun Zhang, Ziming Xu, Ziqiao Xi, Shuang Liang, Ruobing Han, Yuchen Yan, Xinyue Wang, Fan Feng, and Biwei Huang. Causally debiased latent action model for embodied action conditioned world models, 2026.
- [1735] Maurice Weiler and Gabriele Cesa. General $e(2)$ -equivariant steerable CNNs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [1736] William J Welch, Robert J Buck, Jerome Sacks, Henry P Wynn, Toby J Mitchell, and Max D Morris. Screening, predicting, and computer experiments. *Technometrics*, 34(1):15–25, 1992.
- [1737] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011.
- [1738] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches, 2018.
- [1739] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.
- [1740] Jonathan Wenger and Philipp Hennig. Probabilistic linear solvers for machine learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6731–6742. Curran Associates, Inc., 2020.
- [1741] Jonathan Wenger, Geoff Pleiss, Marvin Pförtner, Philipp Hennig, and John Cunningham. Posterior and computational uncertainty in Gaussian processes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 10876–10890. Curran Associates, Inc., 2022.
- [1742] Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10248–10259. PMLR, 2020.
- [1743] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [1744] Jonathan A. Weyn, Dale R. Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002502, 2021. e2021MS002502 2021MS002502.
- [1745] Nick Whiteley, Anthony Lee, and Kari Heine. On the role of interaction in sequential Monte Carlo algorithms. *Bernoulli*, 22(1):494–529, 2016.
- [1746] Kristoffer Wickstrøm, Karl Øyvind Mikalsen, Michael Kampffmeyer, Arthur Revhaug, and Robert Jenssen. Uncertainty-aware deep ensembles for reliable and explainable predictions of clinical time series. *IEEE Journal of Biomedical and Health Informatics*, 25(7):2435–2444, 2020.
- [1747] Jonas Gregor Wiese, Lisa Wimmer, Theodore Papamarkou, Bernd Bischl, Stephan Günnemann, and David Rügamer. Towards efficient MCMC sampling in Bayesian neural networks by exploiting symmetry. In *ECML PKDD*, 2023.

- [1748] Jonas Gregor Wiese, Lisa Wimmer, Theodore Papamarkou, Bernd Bischl, Stephan Guennemann, and David Ruegamer. Towards efficient MCMC sampling in Bayesian neural networks by exploiting symmetry (extended abstract). In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8466–8470. International Joint Conferences on Artificial Intelligence Organization, 2024. Sister Conferences Best Papers.
- [1749] C. K. Wikle, A. Zammit-Mangion, and N. Cressie. *Spatio-Temporal Statistics with R*. CRC Press, 2019.
- [1750] Christopher K. Wikle. Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology*, 84(6):1382–1394, 2003.
- [1751] Christopher K Wikle. Comparison of deep neural networks and deep hierarchical models for spatio-temporal data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(2):175–203, 2019.
- [1752] Christopher K Wikle and Andrew Zammit-Mangion. Statistical deep learning for spatial and spatiotemporal data. *Annual Review of Statistics and Its Application*, 10(1):247–270, 2023.
- [1753] Christopher K Wikle, L Mark Berliner, and Noel Cressie. Hierarchical Bayesian space-time models. *Environmental and ecological statistics*, 5:117–154, 1998.
- [1754] Christopher K Wikle, Ralph F Milliff, Doug Nychka, and L Mark Berliner. Spatiotemporal hierarchical Bayesian modeling tropical ocean surface winds. *Journal of the American Statistical Association*, 96(454):382–397, 2001.
- [1755] Veit David Wild, Robert Hu, and Dino Sejdinovic. Generalized variational inference in function spaces: Gaussian measures meet Bayesian deep learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 3716–3730. Curran Associates, Inc., 2022.
- [1756] Veit David Wild, Sahra Ghalebikesabi, Dino Sejdinovic, and Jeremias Knoblauch. A rigorous link between deep ensembles and (variational) Bayesian methods. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 39782–39811. Curran Associates, Inc., 2023.
- [1757] Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 16837–16864. Curran Associates, Inc., 2023.
- [1758] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks, 2020.
- [1759] Christopher Williams. Computing with infinite networks. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.
- [1760] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.
- [1761] Peter M. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.
- [1762] Adam M. Wilson, Andrew M. Latimer, John A. Silander, Alan E. Gelfand, and Helen de Klerk. A hierarchical Bayesian model of wildfire in a mediterranean biodiversity hotspot: Implications of weather variability and global circulation. *Ecological Modelling*, 221(1):106–112, 2010. Special Issue on Spatial and Temporal Patterns of Wildfires: Models, Theory, and Reality.
- [1763] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1067–1075, Atlanta, Georgia, USA, 2013. PMLR.

- [1764] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (kiss-GP). In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1775–1784, Lille, France, 2015. PMLR.
- [1765] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708. Curran Associates, Inc., 2020.
- [1766] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric Xing. Stochastic variational deep kernel learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1767] Andrew Gordon Wilson. The case for Bayesian deep learning, 2020.
- [1768] Andrew Gordon Wilson. Position: Deep learning is not so mysterious or different. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 82326–82346. PMLR, 2025.
- [1769] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 2016. PMLR.
- [1770] Andrew Gordon Wilson, Pavel Izmailov, Matthew D Hoffman, Yarin Gal, Yingzhen Li, Melanie F Pradier, Sharad Vikram, Andrew Foong, Sanae Lotfi, and Sebastian Farquhar. Evaluating approximate inference in Bayesian deep learning. In Douwe Kiela, Marco Ciccone, and Barbara Caputo, editors, *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pages 113–124. PMLR, 2022.
- [1771] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Response to promises and pitfalls of deep kernel learning, 2025.
- [1772] James T. Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Pathwise conditioning of Gaussian processes. *Journal of Machine Learning Research*, 22(105):1–47, 2021.
- [1773] Lisa Wimmer, Yusuf Sale, Paul Hofman, Bernd Bischl, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures? In Robin J. Evans and Ilya Shpitser, editors, *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 2282–2292. PMLR, 2023.
- [1774] Samuel Wqivst, Jes Frellsen, and Umberto Picchini. Sequential neural posterior and likelihood approximation, 2021.
- [1775] Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [1776] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022.
- [1777] R Gerald Wright, Michael P Murray, and Troy Merrill. Ecoregions as a level of ecological analysis. *Biological conservation*, 86(2):207–213, 1998.
- [1778] Wenyi Wu, Sibozhu, Kun Zhou, Aayush Salvi, Zixuan Song, and Biwei Huang. Structagent: Harness long-horizon digital agents with unified causal structure, 2026.

- [1779] Yi-Shan Wu and Yevgeny Seldin. Split-kl and PAC-Bayes-split-kl inequalities for ternary random variables. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11369–11381. Curran Associates, Inc., 2022.
- [1780] Yi-Shan Wu, Andres Masegosa, Stephan Lorenzen, Christian Igel, and Yevgeny Seldin. Chebyshev-cantelli PAC-Bayes-bennett inequality for the weighted majority vote. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12625–12636. Curran Associates, Inc., 2021.
- [1781] Yi-Shan Wu, Yijie Zhang, Badr-Eddine Chérif-Abdellatif, and Yevgeny Seldin. Recursive PAC-Bayes: A frequentist approach to sequential prior updates with no information loss. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 17947–17971. Curran Associates, Inc., 2024.
- [1782] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation, 2016.
- [1783] Manuel Wüthrich, Sebastian Trimpe, Cristina Garcia Cifuentes, Daniel Kappler, and Stefan Schaal. A new perspective and extension of the Gaussian filter. *The International Journal of Robotics Research*, 35(14): 1731–1749, 2016.
- [1784] Michelle Xia, P. Richard Hahn, and Paul Gustafson. A Bayesian mixture of experts approach to covariate misclassification. *Canadian Journal of Statistics*, 48(4):731–750, 2020.
- [1785] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- [1786] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [1787] Jian Xu, Delu Zeng, and John Paisley. Sparse inducing points in deep Gaussian processes: Enhancing modeling with denoising diffusion variational inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55490–55500. PMLR, 2024.
- [1788] Kai Xu, Hong Ge, Will Tebbutt, Mohamed Tarek, Martin Trapp, and Zoubin Ghahramani. Advancedhmc.jl: A robust, modular and efficient implementation of advanced HMC algorithms. In Cheng Zhang, Francisco Ruiz, Thang Bui, Adjani Bousso Dieng, and Dawen Liang, editors, *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pages 1–10. PMLR, 2020.
- [1789] Wanni Xu, You-Lei Fu, and Dongmei Zhu. ResNet and its application to medical image processing: Research progress and challenges. *Computer Methods and Programs in Biomedicine*, 240:107660, 2023.
- [1790] Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 76806–76838. Curran Associates, Inc., 2023.
- [1791] Adam Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 1812–1842, 2024.
- [1792] Adam X. Yang, Maxime Robeyns, Edward Milsom, Ben Anson, Nandi Schoots, and Laurence Aitchison. A theory of representation learning gives a deep generalisation of kernel methods. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings*

- of the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pages 39380–39415. PMLR, 2023.
- [1793] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report, 2025.
 - [1794] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation, 2019.
 - [1795] Greg Yang and Edward J. Hu. Feature learning in infinite-width neural networks, 2020.
 - [1796] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks, 2019.
 - [1797] Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. SWALP : Stochastic weight averaging in low precision training. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7015–7024. PMLR, 2019.
 - [1798] Jianke Yang, Robin Walters, Nima Dehmamy, and Rose Yu. Generative adversarial symmetry discovery. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39488–39508. PMLR, 2023.
 - [1799] Jianke Yang, Manu Bhat, Bryan Hu, Yadi Cao, Nima Dehmamy, Robin Walters, and Rose Yu. Discovering symbolic differential equations with symmetry invariants, 2025.
 - [1800] Mary Qu Yang, Sherman M. Weissman, William Yang, Jialing Zhang, Allon Canaan, and Renchu Guan. Misc: missing imputation for single-cell RNA sequencing data. *BMC Systems Biology*, 12(7):114, 2018.
 - [1801] Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 64971–64995. Curran Associates, Inc., 2023.
 - [1802] Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. Hierarchical autoregressive modeling for neural video compression, 2020.
 - [1803] Yibo Yang and Stephan Mandt. Towards empirical sandwich bounds on the rate-distortion function. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
 - [1804] Yibo Yang and Stephan Mandt. Computationally-efficient neural image compression with shallow decoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 530–540, 2023.
 - [1805] Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 573–584. Curran Associates, Inc., 2020.
 - [1806] Yibo Yang, Robert Bamler, and Stephan Mandt. Variational Bayesian quantization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10670–10680. PMLR, 2020.
 - [1807] Yibo Yang, Stephan Eckstein, Marcel Nutz, and Stephan Mandt. Estimating the rate-distortion function by Wasserstein gradient descent. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 2768–2794. Curran Associates, Inc., 2023.
 - [1808] Yibo Yang, Stephan Mandt, and Lucas Theis. An introduction to neural data compression. *Foundations and Trends in Computer Graphics and Vision*, 15(2):113–200, 2023.
 - [1809] Yibo Yang, Justus C. Will, and Stephan Mandt. Progressive compression with universally quantized diffusion models, 2024.
 - [1810] W. Yao, Z. Zeng, C. Lian, and H. Tang. Ensembles of echo state networks for time series prediction. In *2013 Sixth International Conference on Advanced Computational Intelligence (ICACI)*, pages 299–304, 2013. 10.1109/ICACI.2013.6748520.

- [1811] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Analysis*, 13(3):917 – 1007, 2018.
- [1812] Yuling Yao, Aki Vehtari, and Andrew Gelman. Stacking for non-mixing Bayesian computations: The curse and blessing of multimodal posteriors. *Journal of Machine Learning Research*, 23(79):1–45, 2022.
- [1813] I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.
- [1814] I-Cheng Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 1998. DOI: <https://doi.org/10.24432/C5PK67>.
- [1815] Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22354–22367. Curran Associates, Inc., 2022.
- [1816] Alec Yenter and Abhishek Verma. Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In *IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, volume 8, 2017.
- [1817] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6613–6623, 2024.
- [1818] Myungsoo Yoo and Christopher K. Wikle. Using echo state networks to inform physical models for fire front propagation. *Spatial Statistics*, 54:100732, 2023.
- [1819] Myungsoo Yoo and Christopher K. Wikle. A Bayesian spatio-temporal level set dynamic model and application to fire front propagation. *The Annals of Applied Statistics*, 18(1):404 – 423, 2024.
- [1820] Myungsoo Yoo, Likun Zhang, and Christopher K Wikle. Real-time forecasting of fire front propagation using the level set method and echo state networks. In *Environmental Modelling with Contemporary Statistics*, pages 265–284. Chapman and Hall/CRC, 2026.
- [1821] Hong-Jun Yoon, Alina Peluso, Eric B Durbin, Xiao-Cheng Wu, Antoinette Stroup, Jennifer Doherty, Stephen Schwartz, Charles Wiggins, Linda Coyle, and Lynne Penberthy. Automatic information extraction from childhood cancer pathology reports. *JAMIA open*, 5(2):ooac049, 2022.
- [1822] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [1823] Hanlin Yu, Marcelo Hartmann, Bernardo Williams Moreno Sanchez, Mark Girolami, and Arto Klami. Riemannian Laplace approximation with the Fisher metric. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 820–828. PMLR, 2024.
- [1824] Hanlin Yu, Arto Klami, Aapo Hyvarinen, Anna Korba, and Omar Chehab. Density ratio estimation with conditional probability paths. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 73146–73174. PMLR, 2025.
- [1825] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020.
- [1826] Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. Dags with no curl: An efficient dag structure learning approach. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12156–12166. PMLR, 2021.

- [1827] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. Cold-start active learning through self-supervised language modeling. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online, 2020. Association for Computational Linguistics.
- [1828] Ming Yuan and Marten Wegkamp. Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11(5):111–130, 2010.
- [1829] Marco Zaffalon. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1):5–21, 2002. Imprecise Probability Models and their Applications.
- [1830] Andrew Zammit-Mangion and Christopher K. Wikle. Deep integro-difference equation models for spatio-temporal forecasting. *Spatial Statistics*, 37:100408, 2020. Frontiers in Spatial and Spatio-temporal Research.
- [1831] Giacomo Zanella. Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, 115(530):852–865, 2020.
- [1832] Jacob Zavatore-Veth and Cengiz Pehlevan. Exact marginal prior distributions of finite Bayesian neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3364–3375. Curran Associates, Inc., 2021.
- [1833] Jacob Zavatore-Veth, Abdulkadir Canatar, Ben Ruben, and Cengiz Pehlevan. Asymptotics of representation learning in finite Bayesian neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24765–24777. Curran Associates, Inc., 2021.
- [1834] Arnold Zellner. Optimal information processing and Bayes’s theorem. *The American Statistician*, 42(4):278–280, 1988.
- [1835] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational Bayes with fixed-point updates, 2020.
- [1836] Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal, 2023.
- [1837] Xueying Zhan, Yaowei Wang, and Antoni B. Chan. Asymptotic optimality for active learning processes. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 2342–2352. PMLR, 2022.
- [1838] Zhang and Lee. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical Science*, 2019.
- [1839] Bai Zhang, David J. Miller, and Yue Wang. Nonlinear system modeling with random matrices: Echo state networks revisited. *IEEE Transactions on Neural Networks and Learning Systems*, 23(1):175–182, 2012.
- [1840] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2019.
- [1841] Cheng Zhang, Stefan Bauer, Paul Bennett, Jiangfeng Gao, Wenbo Gong, Agrin Hilmkil, Joel Jennings, Chao Ma, Tom Minka, Nick Pawlowski, and James Vaughan. Understanding causality with large language models: Feasibility and opportunities, 2023.
- [1842] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.
- [1843] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.
- [1844] Fengshuo Zhang and Chao Gao. Convergence rates of variational posterior distributions. *The Annals of Statistics*, 48(4):2180 – 2207, 2020.

- [1845] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5852–5861. PMLR, 2018.
- [1846] Haoran Zhang, Yunxiao Chen, and Xiaou Li. A note on exploratory item factor analysis by singular value decomposition. *Psychometrika*, 85:358–372, 2020.
- [1847] Jianyi Zhang, Ruiyi Zhang, Lawrence Carin, and Changyou Chen. Stochastic particle-optimization sampling and the non-asymptotic convergence theory. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1877–1887. PMLR, 2020.
- [1848] Jiaqi Zhang, Joel Jennings, Agrin Hilmkil, Nick Pawlowski, Cheng Zhang, and Chao Ma. Towards causal foundation model: on duality between causal inference and attention, 2023.
- [1849] Jiaxin Zhang, Sirui Bi, and Guannan Zhang. A scalable gradient free method for Bayesian experimental design with implicit models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3745–3753. PMLR, 2021.
- [1850] Likun Zhang, Kiran Bhaganagar, and Christopher K. Wikle. Capturing extreme events in turbulence using an extreme variational autoencoder (xvae). *Neural Computing and Applications*, 38(11):428, 2026.
- [1851] Likun Zhang, Xiaoyu Ma, Christopher K. Wikle, and Raphaël Huser. Fast and flexible emulation of spatial extremes processes via variational autoencoders. *Journal of the American Statistical Association*, 0(0):1–14, 2026.
- [1852] Miao Zhang, Steven W. Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12557–12566. PMLR, 2021.
- [1853] Ruqi Zhang, A. Feder Cooper, and Christopher De Sa. Asymptotically optimal exact minibatch metropolis-hastings. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19500–19510. Curran Associates, Inc., 2020.
- [1854] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [1855] Ruqi Zhang, Xingchao Liu, and Qiang Liu. A Langevin-like sampler for discrete distributions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26375–26396. PMLR, 2022.
- [1856] Ruqi Zhang, Andrew Gordon Wilson, and Christopher De Sa. Low-precision stochastic gradient Langevin dynamics. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26624–26644. PMLR, 2022.
- [1857] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [1858] Tong Zhang. Theoretical analysis of a class of randomized regularization methods. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 156–163, 1999.
- [1859] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Conference on Machine Learning*, 2000.

- [1860] Wenxuan Zhang, Youssef Mohamed, Bernard Ghanem, Philip Torr, Adel Bibi, and Mohamed Elhoseiny. Continual learning on a diet: Learning from sparsely labeled streams under constrained computation. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 31516–31532, 2024.
- [1861] Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. Limix: Unleashing structured-data modeling capability for generalist intelligence, 2025.
- [1862] Yangfan Zhang and Yun Yang. Bayesian model selection via mean-field variational approximation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(3):742–770, 2024.
- [1863] Zhiwang Zhang, Hongliang Sun, Shuqing Li, Jing He, Jie Cao, Guanghai Cui, and Gang Wang. Two-stage sparse multi-kernel optimization classifier method for more accurate and explainable prediction. *Expert Systems with Applications*, 231:120635, 2023.
- [1864] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Symmetry teleportation for accelerated optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16679–16690. Curran Associates, Inc., 2022.
- [1865] He Zhao, Piyush Rai, Lan Du, Wray Buntine, Dinh Phung, and Mingyuan Zhou. Variational autoencoders for sparse and overdispersed discrete data. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1684–1694. PMLR, 2020.
- [1866] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.
- [1867] Zheng Zhao, Muhammad Emzir, and Simo Särkkä. Deep state-space Gaussian processes. *Statistics and Computing*, 31(6):75, 2021.
- [1868] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric Xing. Dags with no tears: Continuous optimization for structure learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [1869] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization, 2020.
- [1870] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.
- [1871] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62307–62331. PMLR, 2024.
- [1872] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 83214–83247, 2025.
- [1873] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2016.
- [1874] Xingyu Zhou, Yuling Jiao, Jin Liu, and Jian Huang. A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, 118(543):1837–1848, 2023.

- [1875] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 7103–7114. Curran Associates, Inc., 2022.
- [1876] Yuren Zhou, Yuqi Gu, and David B. Dunson. Bayesian deep generative models for multiplex networks with multiscale overlapping clusters, 2024.
- [1877] Yuren Zhou, Yuqi Gu, and David B. Dunson. Bayesian latent class regression with interpretable binary profiles, 2025.
- [1878] Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4923–4932, 2019.
- [1879] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58–65, 2003.
- [1880] Yifan Zhu, Pranay Thangeda, Melkior Ornik, and Kris Hauser. Few-shot adaptation for manipulating granular materials under domain shift, 2023.
- [1881] Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [1882] Jingwei Zhuo, Chang Liu, Jiabin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing Stein variational gradient descent. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 6018–6027. PMLR, 2018.
- [1883] Alessandro Zito, Tommaso Rigon, and David B Dunson. Inferring taxonomic placement from DNA bar-coding aiding in discovery of new taxa. *Methods in Ecology and Evolution*, 14(2):529–542, 2023.
- [1884] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
- [1885] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2016.
- [1886] Difan Zou, Pan Xu, and Quanquan Gu. Stochastic gradient Hamiltonian Monte Carlo methods with recursive variance reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.