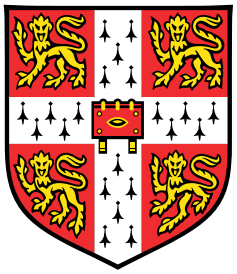
Scalable Bayesian Inference in the Era of Deep Learning

From Gaussian Processes to Deep Neural Networks



Javier Antorán Cabiscol

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Darwin College

May 2024

I would like to dedicate this thesis to the memory of my grandmother, Carmen Mir Gavara, who kept reminding me that I had to submit my thesis.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Javier Antorán Cabiscol
May 2024

Acknowledgements

First and foremost, I would like to thank my supervisor Miguel Hernández-Lobato. Miguel gave me the opportunity to pursue a PhD at a time when I was unsure what next step to take professionally. Throughout the PhD, Miguel has given me complete freedom to pursue my interests and to collaborate freely with other researchers. I have learnt a lot from Miguel's capacity to boil down complex topics to their simplest form and from his optimistic outlook on research. I would also like to thank him for his patience throughout all of the times I ignored his advice and went on to try things that didn't work just to find out that his previous suggestion indeed was the best way forward.

I have been very lucky to have worked with excellent collaborators during my PhD. There is a relatively widespread bias in academia by which most of the credit for research publications is assigned to the first author. This creates perverse incentives by which secondary authors are discouraged from making significant contributions to collaborative projects. In my experience, the easiest way to perform great research is to have multiple talented collaborators fully committed to a project. Indeed, most of my work, and definitely my best work, has been co-first authored with James Allingham, Riccardo Barbano, Shreyas Padhi, and Andy Lin. Apart from helping me escape the academic credit assignment trap, James has been a great friend. He is likely the person with whom I have achieved best working synergy within my professional career. I will always regret not having worked more closely together during the later stages of our PhDs. Riccardo Barbano is another good friend with whom I was privileged to work closely. Riccardo introduced me to the world of computed tomography, resulting in the final content chapter of this thesis. I started working with Shreyas and Andy in the last two years of my PhD, which allowed me to play a more senior role in our collaboration. I learned a lot from this arrangement, and watching them both grow into excellent researchers has felt very rewarding. I would be remiss to not also give special mention to Dave Janz, who has accompanied me throughout my research into linearised Laplace and Gaussian processes. I have learned a lot of maths from Dave and I am very grateful to him for his patience when teaching me new concepts.

I would also like to thank a number of additional collaborators: Alex Terenin, who introduced me to matrix-free linear algebra and from whom I learned a lot about academic writing, Laurence Midgeley, whose enthusiasm and endless stream of clever ideas are inspiring, Erik Daxberger, Umang Bhatt, Johannes Leuschner, Austin Tripp, Vincent Stimper, Emile Mathieu, Tomas Geffner, Adam Foster, Wenbo Gong, Chao Ma, Chelsea Murray, Zeljko Kereta, Tameem Adel, Adrian Weller, Bernhard Schölkopf, Bangti Jin, Csaba Szepesvári, and Eric Nalisnick. Apart from being a great collaborator, I am grateful to Eric for, together with Max Welling, hosting me during my visit to AMLab at the the university of Amsterdam. I am also grateful to Artem Artemev and Mark Van der Wilk for our very insightful conversations.

I would like to thank Marine Schimel for patiently putting up with me during the highs and lows of the PhD and also to Manuel Escolá, Cristina Uruén, Laura Aznar, Beatriz Alegre, Juan Galvez, Pedro Cabeza, David del Río, Marta Parra, Gergely Flamich, Stratis Markou, and Miguel García Ortegon, for being good friends. Additionally, I am grateful to Adriá Garriga Alonso, Andrew Foong, Kris Jensen, Sebastian Ober, Matt Ashman, Tor Erlend Fjelde, Adrían Goldwaser, Juyeon Hao, Bruno Mlodozeniec, Kenza Tazi, Jonny So, Valerii Likhosherstov, Aliaksandra Shysheya, Vincent Dutordoir, Runa Eschenhagen, Emile Mathieu, Will Tebutt, and Isaac Reid for making the CBL a nice environment in which to have spent the past four years.

I am grateful to Yann Dubois for being the only person I know who shares my obsessive passion for machine learning, and to Antonio Miguel for very generously dedicating endless hours to teaching me machine learning during my years as an undergraduate student.

Finally, I would like to thank my parents who gave me every opportunity.

My PhD research has been supported by Microsoft Research, through its PhD Scholarship Programme, and by the EPSRC. My work was also supported by a number of Tier-2 capital grants that allowed me access to the University of Cambridge Research Computing Services. I apologise to the Cambridge HPC staff for taking down the queuing server by submitting too many jobs one time.

Abstract

Large neural networks trained on large datasets have become the dominant paradigm in machine learning. These systems rely on maximum likelihood point estimates of their parameters, precluding them from expressing model uncertainty. This may result in overconfident predictions and it prevents the use of deep learning models for sequential decision making.

This thesis develops scalable methods to equip neural networks with model uncertainty. To achieve this, we do not try to fight progress in deep learning but instead borrow ideas from this field to make probabilistic methods more scalable. In particular, we leverage the linearised Laplace approximation to equip pre-trained neural networks with the uncertainty estimates provided by their tangent linear models. This turns the problem of Bayesian inference in neural networks into one of Bayesian inference in conjugate Gaussian-linear models. Alas, the cost of this remains cubic in either the number of network parameters or in the number of observations times output dimensions. By assumption, neither are tractable.

We address this intractability by using stochastic gradient descent (SGD)—the workhorse algorithm of deep learning—to perform posterior sampling in linear models and their convex duals: Gaussian processes. With this, we turn back to linearised neural networks, finding the linearised Laplace approximation to present a number of incompatibilities with modern deep learning practices—namely, stochastic optimisation, early stopping and normalisation layers—when used for hyperparameter learning. We resolve these and construct a sample-based EM algorithm for scalable hyperparameter learning with linearised neural networks.

We apply the above methods to perform linearised neural network inference with ResNet-50 (25M parameters) trained on Imagenet (1.2M observations and 1000 output dimensions). To the best of our knowledge, this is the first time Bayesian inference has been performed in this real-world-scaled setting without assuming some degree of independence across network weights. Additionally, we apply our methods to estimate uncertainty for 3d tomographic reconstructions obtained with the deep image prior network, also a first. We conclude by using the linearised deep image prior to adaptively choose sequences of scanning angles that produce higher quality tomographic reconstructions while applying less radiation dosage.

Table of contents

Nomenclature

Acronyms / Abbreviations

AD Automatic Differentiation

AI Artificial Intelligence

CG Conjugate Gradient

CNN Convolutional Neural Network

CT Computed Tomography

DIP Deep Image Prior

$ELBO$ Evidence Lower Bound

EP expectation propagation

GGN Generalised Gauss Newton Matrix

GP Gaussian Processes

GP Gaussian Processes

HMC Hamiltonian Monte Carlo

LL Log Likelihood

MAP Maximum A Posteriori

MC Monte Carlo

$MCMC$ Markov Chain Monte Carlo

μCT Micro Computed Tomography

NLL Negative Log Likelihood

NTK Neural Tangent Kernel

OL Online Laplace

PD Positive Definite

$\mathcal{P}redCP$ Predictive Complexity Prior

PSD Positive Semidefinite

RBF Radial Basis Function

$RKHS$ Reproducing Kernel Hilbert Space

SDD Stochastic Dual Descent

SGD Stochastic Gradient Descent

$SVGP$ Stochastic Variational Gaussian Process

TV Total Variation

VI Variational Inference

Chapter 1

Introduction

Programs learnt from data are rapidly displacing programs based on human-designed rules as the dominant paradigm for computer-based automation. We have seen this in the fields of computer vision (Dosovitskiy et al., 2021), inverse problems (Arridge et al., 2019), natural language processing (Wang et al., 2024), information retrieval (Zhu et al., 2024), text and image generation (Jiang et al., 2024; Saharia et al., 2022), system control (Hu et al., 2022), scientific discovery (Collaboration et al., 2021; Graczykowski et al., 2022), and even computer programming (Chen et al., 2021), among others. Practically all of these advances were enabled by large-scale deep learning (Henighan et al., 2020). Indeed, it is plausible that given enough data, a flexible enough neural network, and sufficient compute to train the artificial intelligence (AI), data-driven decision making methods will dominate all traditional computer programs.

The rules for optimally learning from data were codified in the framework of Bayesian probability well before the deep learning revolution of the past decade (Cox, 1946; Jaynes and Justice, 1986; Jeffreys, 1939; Stigler, 1986). Under this framework, we represent our knowledge, or lack thereof, as probability distributions. When we observe new data, the information gained is used to update these prior distributions into less entropic posterior distributions (Gull, 1988; Skilling, 1989). In turn, these act as priors for future inferences. Although probabilistic methods were extensively leveraged to build primordial neural network systems (Hinton and van Camp, 1993; Salakhutdinov and Hinton, 2009), modern neural network methods rely on expressing our beliefs as point estimates instead of probability distributions. The lack of explicitly modelled uncertainty makes modern deep learning systems vulnerable to acting spuriously when they encounter situations which were not provided sufficient coverage in the training data (Goddard, 2023; Weiser and Schweber, 2023).

Additionally, probabilistic methods remain state of the art for decision-making tasks that require uncertainty-based exploration, like automated chemical design (Gómez-Bombarelli et al., 2018).

From a Bayesian perspective, neural networks can be seen as an uncompromising model choice that puts very little restrictions on the function class to be learnt. The effects of individual weights are non-interpretable, precluding the design of informative Bayesian priors for neural network parameters. However, it is likely this is the very feature that allows us to use neural networks to solve tasks in ways that can not easily be summarised by a human-readable list of rules. For instance, how to eloquently sustain a conversation or drive a car. With this idea in place, an intuitive way to explain the seeming incompatibility between Bayesian inference an neural networks is to think of the former as scoring a set of prior hypotheses by how well each one it agrees with the data. The problem with modern neural networks is that there are just too many hypotheses to score. The scoring becomes prohibitively expensive, especially, when combined with large datasets which are likely to be fit well by a relatively small region of the neural network parameter space. In other words, while maximum likelihood learning scales well to the modern big-network and big-data setting, Bayesian inference does not.

This thesis aims to bridge the gap between Bayesian methods and contemporary deep learning. This endeavour was pioneered by Mackay (1992a) who extended Bayesian inference and hyperparameter selection in linear models (which is also attributable to Gull (1989)) to the neural network setting via the Laplace approximation, naming his class of methods the *evidence framework*. In the last 30 years, the methods of machine learning have changed quite a bit; the scale of the problems tackled and models deployed has grown by multiple orders of magnitude, precluding the out-of-the-box application MacKay's methods, and giving me something to write my thesis about. In fact, similarly to MacKay (1992a), this thesis begins by making contributions to the field of linear models and Gaussian processes, uses the Laplace approximation to adapt these methods for approximate inference in neural networks, and finally applies the developed Bayesian neural networks to efficient data acquisition. Thus, this thesis is perhaps best described as a modern take on the evidence framework which makes it scalable to modern problem sizes and amenable to modern deep learning architectures.

To achieve our goals, we are not going to fight progress in deep learning by trying to re-build it from the ground up to natively use Bayesian inference, for instance by imposing fancy handcrafted priors on weights whose effect we dont understand. I believe this is a lost cause. Instead, we are going to build upon the tremendous progress that has been made in deep learning, and borrow ideas from this field to make Bayesian methods more scalable.

For instance, in Chapter 4, we will use stochastic gradient descent—the de-facto method for training neural networks—to make Bayesian inference in linear models and Gaussian processes more scalable. Additionally, when dealing with neural networks, we will focus on the *post-hoc inference* setting, in which we leverage approximate Bayesian methods, to obtain uncertainty estimates for pre-trained neural networks. This will ensure the thesis' contributions remain compatible with the quickly evolving field of deep learning.

1.1 Thesis outline and contributions

This thesis is written with my past self, before embarking on the PhD, as a target audience. Although some measure theoretic and functional analytic concepts are (infrequently) mentioned throughout the thesis, knowledge of these fields is not required to understand the thesis' contributions. Additionally, I have tried to combine mathematical derivations with a number of less-technical remarks to help the reader build intuition about the material.

The rest of this thesis is organised as follows.

- Chapter 2 introduces Bayesian inference in conjugate Gaussian-linear models and Gaussian processes. Particular focus is placed on the duality between the two model classes because we will make heavy use of it throughout the thesis. We also introduce the model evidence and discuss hyperparameter learning in linear models. Readers who posses an adept grasp of Bayesian linear models may skip this chapter. However, Section 2.3 on pathwise conditioning may still be of interest, since it is slightly more niche.

- Chapter 3 introduces approximate inference for large scale or non-conjugate linear models, and for neural networks. We discuss variational inference, conjugate gradient-based approximate inference, and the Laplace approximation. We discuss the limitations of each of these approximations emphasising the trade-off between crudeness of approximation and scalability that they all present. Special focus is placed on the linearised variant of the Laplace approximation, as its adaption to modern deep learning will be a key theme of this thesis.

- Chapter 4 leverages stochastic gradient descent to scale Bayesian inference in conjugate linear models and Gaussian processes to large scale problems. In particular, we develop a number of quadratic objectives whose minimisers represent samples from the posterior distribution of a Gaussian process. We analyse their properties and propose

a series of recommendations on how best to apply stochastic gradient-based solvers to this setting. We dub our approach stochastic dual descent. It presents a linear computational cost in the number of observations per gradient step, and we find that a constant number of steps suffice to obtain good performance across a diverse range of problem settings. This starkly contrasts with the cubic in the number of observations cost of exact inference. We also extend stochastic gradient descent to inducing point posteriors, where a sub-linear cost per step can be achieved. We analyse the spectral bias of solutions found via stochastic gradient descent, showing that full convergence is not necessary to achieve strong performance. Experimentally, we show that stochastic dual descent outperforms conjugate gradient-based inference and variational inference on standard regression benchmarks and on a large-scale Bayesian optimisation benchmark. When combined with stochastic dual descent, Gaussian processes are able match the performance of graph neural networks on a large scale molecular binding affinity prediction task. This chapter is based on Antorán et al. (2023), Lin et al. (2023b) and Lin et al. (2024).

- Chapter 5 identifies a number of incompatibilities between the classical linearised Laplace model evidence objective for model selection and modern deep learning methodologies, in particular, stochastic optimisation, early stopping, and the use of normalisation layers. These result in severe deterioration of the model evidence estimate. We provide recommendations on how to adapt linearised Laplace in light of these issues. Namely, every neural network weight setting has an associated tangent linear model, and we recommend using the evidence of this linear model to select the hyperparameters to be used for linearised Laplace uncertainty estimation. Additionally, we must select priors over the linearised network weights which counteract a number of scale invariances introduced by normalisation layers. We empirically validate our recommendations on MLPs, classic CNNs, residual networks with and without normalisation layers, generative autoencoders and transformers. This chapter is based on Antorán et al. (2022) and Antorán et al. (2022).

- Chapter 6 combines the contributions of the previous two chapters to put forth a scalable sample-based EM algorithm for hyperparameter learning in linearised neural networks. The E-step is based on stochastic-gradient descent posterior sampling and the M-step leverages a sample-based estimate of the effective dimension-based hyperparameter update introduced by Mackay (1992a). We also discuss a number of implementation details that allow us to work with linearised neural networks without ever instantiating these models' Jacobians explicitly. This suite of techniques allows us to scale linearised

neural network inference to modern architectures and datasets, such as ResNet-50 on Imagenet. Distinctly from more crude, e.g. factorised, approximations to Bayesian inference in Neural networks, our approach improves upon the performance of the pre-trained network we build upon. It provides state of the art results in terms of joint predictions across multiple inputs, a task of special interest for uncertainty-guided exploration. This chapter is based on Antorán et al. (2023).

- Chapter 7 applies the methods developed in this thesis to uncertainty estimation and experimental design for computed tomography (CT) image and volume reconstruction. In particular, we use the deep image prior architecture for reconstruction and linearise the network for uncertainty estimation. We develop a novel total-variation based prior for the linearised deep image prior. Our scalable sample-based EM iteration allows our method to scale to high-resolution 3d volumetric reconstructions from real-measured micro CT data. To the best of our knowledge, our work is the first to perform uncertainty estimation for 3d neural reconstructions. We then go on to leverage the linearised deep image prior as a data-dependent prior for adaptive CT scanning angle selection. This allows us to design strategies that reduce by up to 30% the number of scans needed to match the performance of an equidistant angle baseline on a synthetic task. This chapter is based on Barbano et al. (2022a), Barbano et al. (2022b), Antoran et al. (2023) and Antorán et al. (2023).

- Chapter 8 concludes the thesis with an outlook of this thesis' contributions in the context of the broader field of machine learning and a discussion of avenues for future work.

1.2 Full list of publications

I now provide a full list of papers I have written during my time as a PhD student. Titles are bolded for papers whose content is included in this thesis. I also give a brief description of my contribution to each of these works. An asterisk superscript * denotes co-first authorship.

1. J.A. Lin*, S. Padhy*, **J. Antorán***, A. Tripp, A. Terenin, C. Szepesvári, J. M. Hernández-Lobato, D. Janz. "**Stochastic Gradient Descent for Gaussian Processes Done Right.**" In *International Conference on Learning Representations (ICLR)*. 2024

   My contribution to to this project consisted of developing the idea, helping my co-authors debug their code, writing the paper, and helping orchestrate other authors' contributions.

2. L.I. Midgley*, V. Stimper*, **J. Antorán***, E. Mathieu*, B. Schölkopf, Hernández-Lobato. "SE (3) equivariant augmented coupling flows." In *Advances in Neural Information Processing Systems (NeurIPS)*. 2023. Awarded spotlight presentation.

3. J.A. Lin, **J. Antorán**, J.M. Hernández-Lobato. "Online Laplace Model Selection Revisited." In *Symposium on Advances in Approximate Bayesian Inference (AABI)*. 2023. Awarded oral presentation.

4. J.A. Lin*, **J. Antorán***, S. Padhy*, D. Janz, J.M. Hernández-Lobato, A. Terenin. "**Sampling from Gaussian Process Posteriors using Stochastic Gradient Descent**." In *Advances in Neural Information Processing Systems (NeurIPS)*. 2023. Awarded oral presentation.

   My contribution to to this project consisted of developing the idea, writing some of the code, writing the paper, and helping orchestrate other authors' contributions.

5. R. Barbano, **J. Antorán**, J. Leuschner, J.M. Hernández-Lobato, B. Jin, Z. Kereta. "Image Reconstruction via Deep Image Prior Subspaces." In *Transactions on Machine Learning Research (TMLR)*. 2023

6. J.U. Allingham, **J. Antorán**, S. Padhy, E. Nalisnick, J.M. Hernández-Lobato. "Learning Generative Models with Invariance to Symmetries." In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*. 2022 Allingham et al. (2022)

7. **J. Antorán***, S. Padhy*, R. Barbano, E. Nalisnick, D. Janz, J. M. Hernández-Lobato. "**Sampling-based inference for large linear models, with application to linearised Laplace.**" In *International Conference on Learning Representations (ICLR)*. 2023

   My contribution to to this project consisted of developing the idea, writing the codebase that was used for experiments, running some experiments, writing the paper, and helping orchestrate other authors' contributions.

8. **J. Antorán***, R. Barbano*, J. Leuschner, J.M. Hernández-Lobato, B. Jin. "**Uncertainty Estimation for Computed Tomography with a Linearised Deep Image Prior.**" In *Transactions on Machine Learning Research (TMLR)*. 2023

   My contribution to to this project consisted of developing the idea, helping my co-authors debug their code, and writing the paper.

9. R. Barbano*, J. Leuschner*, **J. Antorán***, B. Jin, J.M. Hernández-Lobato. "**Bayesian experimental design for computed tomography with the linearised deep image prior.**" In *Adaptive Experimental Design and Active Learning workshop at ICML*. 2022

My contribution to to this project consisted of developing the idea, helping my co-authors debug their code, and writing the paper.

10. **J. Antorán**, D. Janz, J.U. Allingham, E. Daxberger, R.R. Barbano, E. Nalisnick, J. M. Hernández-Lobato. "**Adapting the linearised Laplace model evidence for modern deep learning.**" In *International Conference on Machine Learning*. 2022

    My contribution to to this project consisted of developing the idea, writing the codebase that was used for the experiments, writing the paper, and orchestrating other co-authors' contributions.

11. C. Murray, J.U. Allingham, **J. Antorán**, J.M. Hernández-Lobato. "Addressing bias in active learning with depth uncertainty networks... or not." In *Proceedings of Machine Learning Research (PMLR) 163:59-63*. 2022

12. T. Geffner*, **J. Antorán***, A. Foster*, W. Gong, C. Ma, E. Kiciman, A. Sharma, A. Lamb, M. Kukla, N. Pawlowski, M. Allamanis, C. Zhang. "Deep end-to-end causal inference." *arXiv preprint arXiv:2202.02195*. 2022

13. **J. Antorán**, J.U. Allingham, D. Janz, E. Daxberger, E. Nalisnick, J. M. Hernández-Lobato. "**Linearised Laplace inference in networks with normalisation layers and the neural g-prior.**" In *Symposium on Advances in Approximate Bayesian Inference (AABI)*. 2022. Awarded oral presentation.

    My contribution to to this project consisted of developing the idea, writing the codebase that was used for the experiments, writing the paper, and orchestrating other co-authors' contributions.

14. R. Barbano*, **J. Antorán***, J.M. Hernández-Lobato, B. Jin. "**A probabilistic deep image prior over image space.**" In *Symposium on Advances in Approximate Bayesian Inference (AABI)*. 2022

    My contribution to to this project consisted of developing the idea, helping my co-authors debug their code, and writing the paper.

15. C. Murray, J.U. Allingham, **J. Antorán**, J.M. Hernández-Lobato. "Depth Uncertainty Networks for Active Learning." In *Bayesian Deep Learning Workshop at the 35th Conference on Neural Information Processing System*. 2021

16. U. Bhatt, **J. Antorán**, Y. Zhang, Q.V. Liao, P. Sattigeri, R. Fogliato, G. G. Melancon, R. Krishnan, J. Stanley, O. Tickoo, L. Nachman, R. Chunara, A. Weller, A. Xiang.

"Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty." In *AAAI/ACM Conference on AI, Ethics, and Society*. 2021

17. E. Daxberger, E. Nalisnick, J.U. Allingham, **J. Antorán**, J.M. Hernández-Lobato. "Bayesian Deep Learning via Subnetwork Inference." In *International Conference on Machine Learning (ICML)*, 2021.

18. **J. Antorán**, U. Bhatt, T. Adel, A. Weller, J. M. Hernández-Lobato. "Getting a CLUE: A Method for Explaining Uncertainty Estimates." In *International Conference on Learning Representations (ICLR).* 2021. Awarded oral presentation.

19. E. Daxberger, E. Nalisnick, J. Allingham, **J. Antorán**, J.M. Hernández-Lobato. "Expressive yet tractable Bayesian deep learning via subnetwork inference." *Symposium on Advances in Approximate Bayesian Inference (AABI)* 2020. Awarded oral presentation.

20. **J. Antorán***, J.U. Allingham*, J.M. Hernández-Lobato. "Depth uncertainty in neural networks." In *Advances in Neural Information Processing Systems (NeurIPS)*. 2020

21. **J. Antorán***, J. U. Allingham*, J. M. Hernández-Lobato. "Variational depth search in ResNets." In *Workshop on Neural Architecture Search at International Conference on Learning Representations*. 2020

Chapter 2

Bayesian reasoning with Gaussian linear models and Gaussian processes

We start with *linear regression*, where outputs are given by linear functions of some basis function expansion of the input variables, as these models play a central role in this thesis. When a Gaussian prior is placed over the parameters and the targets are assumed to have been corrupted by additive Gaussian noise, we obtain the Gaussian linear model. This setting is of special interesting because conjugacy between likelihood and prior leads to the equations of Bayesian inference admitting closed form solutions. This simplicity does not come at the cost of flexibility; the use of basis function expansion allows linear regressors to learn arbitrarily complex functions. This thesis will leverage this fact to tackle the analytical intractability of Bayesian inference in neural network models; in Chapter 6 we will approximate the predictions of the neural network with those of a Gaussian linear model with an appropriate choice of basis function expansion. The key limitation of Gaussian linear regression is its computational cost, which scales cubically with the number of observations or number of model parameters. This thesis addresses this limitation in Chapter 4.

I would be remiss to not mention some other excellent references for Gaussian linear models, such as the seminal texts of Gull (1989) and MacKay (1992b), and the books of Bishop (2006) (Chapter 3) and Williams and Rasmussen (2006) (Chapter 2). However, this chapter provides a presentation of the material that emphasises the duality between parameter-space and function-space, and the pathwise formulation of inference, which will hopefully make the contributions of the rest of the thesis easily accessible to the reader. In particular, we start by providing 3 complementary views of Bayesian inference in Gaussian-linear models: Section 2.1 introduces the parametric weight-space view of linear

regression, Section 2.2 introduces Gaussian processes (GP) , the dual, non-parametric view of linear regression, and Section 2.3 presents the pathwise view of inference in Gaussian processes which deals directly with random functions. The latter will be key to designing computationally efficient inference algorithms in Chapter 4. We then go on to discuss the importance of the choice of hyperparameters for linear models and how to select them via marginal likelihood maximisation in Section 2.4. The chapter concludes with a discussion of the limitations of linear models in Section 2.5.

2.1 The weight space view: Gaussian linear regression

We begin by introducing a multi-output basis-function linear model. We observe a set of n inputs $x_1, \ldots, x_n \in \mathcal{X}$ and corresponding outputs $y_1, \ldots, y_n \in \mathcal{Y} \subseteq \mathbb{R}^c$, where c is the number of output dimensions. We introduce a basis function expansion $\phi : \mathcal{X} \to \mathbb{R}^c \times \mathcal{H}$, that maps inputs into some Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. We will not provide a review of RKHS, but instead we refer the reader to Paulsen and Raghupathi (2016) for this. Since we will always work with separable RKHS, we can treat their elements as vectors, that is $\mathcal{H} \subseteq \mathbb{R}^d$, without loss of generality—although sometimes these vectors will be infinite dimensional—and thus we henceforth treat $\phi(x_i)$ like a $c \times d$ dimensional matrix.

We assume that the targets are generated as a linear combination of our featurised inputs, weighted by a parameter vector $w \in \mathcal{H} \subset \mathbb{R}^d$, and corrupted by additive Gaussian noise with 0 mean and observation-dependent positive definite (PD) and symmetric precision matrix $B_i \in \mathbb{R}^{c \times c}$ for each $i \leq n$. That is, each target is given by

$$y_i = \phi(x_i)w + \varepsilon_i \quad \text{with} \quad w \sim \mathcal{N}(0, A^{-1}) \quad \text{and} \quad \varepsilon_i \sim \mathcal{N}(0, B_i^{-1}). \tag{2.1}$$

The parameter vector w is an unobserved variable which we assume to have been drawn from a Gaussian prior distribution with precision given by the positive definite matrix A.

Henceforth, we will use the following stacked notation: we write $Y \in \mathbb{R}^{nc}$ for the concatenation of $y_1, \ldots, y_n$. We stack the expanded observations into the design matrix $\Phi = [\phi(x_1)^T, \phi(x_2)^T, \ldots, \phi(x_n)^T]^T \in \mathbb{R}^{nc \times d}$. We concatenate the additive noise vectors into $\mathcal{E} = [\varepsilon_1^T, \varepsilon_2^T, \ldots, \varepsilon_n^T]^T$. Its distribution is a zero centred Gaussian with $B \in \mathbb{R}^{nc \times nc}$, the block diagonal matrix with blocks $B_1, \ldots, B_n$, as its precision. With this, our model is

$$Y = \Phi w + \mathcal{E} \quad \text{with} \quad w \sim \mathcal{N}(0, A^{-1}I) \quad \text{and} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}). \tag{2.2}$$

Without loss of generality we assume $A = aI$ with $a \in \mathbb{R}_+$; any additional structure in A can be absorbed into the basis functions ϕ. Additionally, unless specified otherwise, we assume isotropic observation noise $B = bI$ with $b \in \mathbb{R}_+$. With this, each output dimension can be seen as an additional independent observation, and we are free to assume $c = 1$ without loss of generality. Finally, for a vector v and a Positive SemiDefinite (PSD) matrix G of compatible dimensions, $\|v\|_G^2 = v^T G v$.

Remark On our construction of the multioutput linear model
Our model generates multiple outputs from a weight vector by multiplying with the matrix-valued features $\phi(x_i)$. This choice differs from the presentation of Bishop (2006), where a weight matrix multiplies vector-valued features. Our choice is deliberate, as it will simplify notation when dealing linear approximations to multioutput neural network functions in later chapters.

Notation for probability distributions We use capital letters to refer to probability measures, e.g., $\Pi = \mathcal{N}(0, A^{-1})$ and lowercase letters to refer to their density functions, i.e. π. These are defined in the standard way via the Radon–Nikodym derivative $d\Pi = \pi(w)d\nu$ with ν denoting the Lebesgue measure. We will not concern ourselves any further with measure theoretic issues. Throughout the thesis we assume any necessary conditions hold for simplicity. We refer to the parameters of probability distributions which we do not treat probabilistically as hyperparameters. To make these explicit in our notation for a density function, we separate the hyperparameters from the point in the sample space at which the density is evaluated by a semicolon. For instance, we may write the density of our Gaussian prior at w as $\pi(w;\, 0, A)$ where the mean 0 and precision A are the hyperparameters. However, whenever there is no ambiguity, we omit the hyperparameters to keep our notation uncluttered. We write conditional density functions, e.g. likelihood functions, by separating variables being conditioned on with a vertical bar $|$. For instance, the likelihood of the linear regression weights is written as $p(Y|w) = \frac{\partial P_{Y|w}}{\partial \nu}$, where $P_{Y|w} = \mathcal{N}(\Phi w,\, B^{-1})$. When there is no ambiguity, we do not explicitly include our set of inputs X in our notation to further reduce clutter. Finally, we will also refer to the density of a distribution by prepending the point at which it being evaluated to the distribution's arguments, but separated by a semicolon. That is, $\mathcal{N}(w;\, 0, A^{-1}) = \pi(w)$.

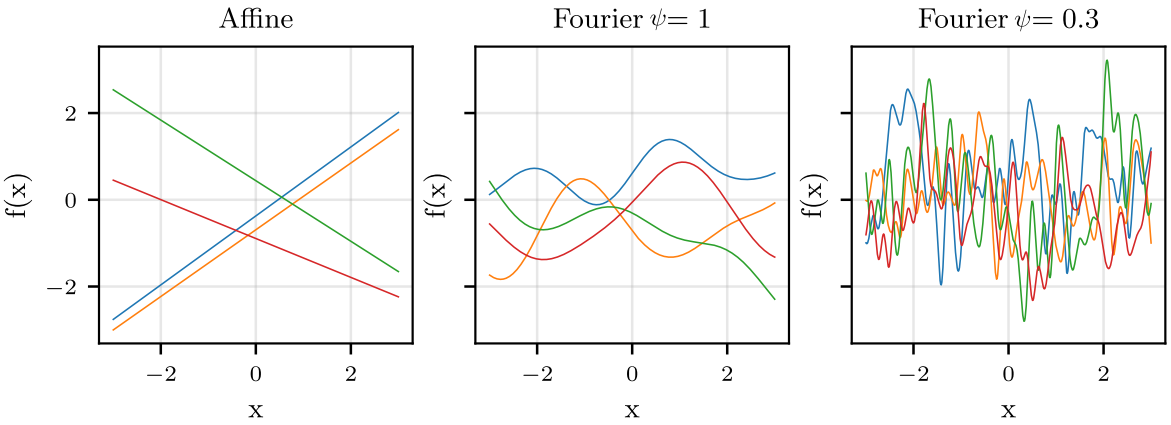


Fig. 2.1 Each plot displays four prior function samples, drawn using (2.5). The left side plot uses an affine basis expansion (2.3), the middle one a 500 element random Fourier expansion with a Gaussian spectral measure and a lengthscale of $\psi = 1$ (2.4), and the right side plot uses a similar Fourier expansion but with a lengthscale of $\psi = 0.3$.

2.1.1 Understanding our choice of model

The choice of basis is perhaps the most important modelling decision when working with linear models; our flexibility in the choice of basis makes linear models very powerful. Indeed, every function can be expressed as a linear combination of a set of basis functions; to see this just choose an element of the basis to contain the target function. However, we seek more than just a representation from which our targets can be linearly decoded[1]. Our basis should reflect our prior knowledge (and uncertainty) over the target function.

To illustrate the power of the basis function expansion, we provide some examples of common basis function choices: the affine basis and the random Fourier basis. We restrict ourselves to $\mathcal{X} = \mathbb{R}$ and a single output dimension $c = 1$ for the purpose of visualisation. The affine basis corresponds to regression with a single linear weight and a bias. That is

$$\phi(x) = [1, x]. \tag{2.3}$$

This model expresses the belief that our target function is a straight line, or plane. Furthermore, within the set of all possible lines, our 0-centred Gaussian prior over the parameters w expresses a belief that lines corresponding to weight and bias choices of small magnitude are more likely a priori. The random Fourier basis (Rahimi and Recht, 2007; Sutherland and Schneider,

[1]Many trivial choices of basis, like ones with very short lengthscales, allow any target to be linearly decoded but are not practically useful.

2015) represents the input as a set of cosines with random frequency and phase

$$\phi_{s,r}(x) = \sqrt{\frac{2}{d}}[\cos(s_1^T x + r_1), \cos(s_2^T x + r_2), \ldots, \cos(s_d^T x + r_d)] \tag{2.4}$$
$$\text{with} \quad s_i \sim \mathcal{N}(0, \psi^{-2}) \qquad \text{and} \qquad r_i \sim \text{Uniform}(0, 2\pi),$$

where the subscript in $\phi_{s,r}$ makes explicit the features dependence on the source of randomness s, r. The lengthscale parameter ψ controls the smoothness of the functions we can express through the choice of frequency variance. Small values lead to our prior placing most of weight on smooth functions, while large values generate a mix of functions of different smoothness.

We use $f : \mathcal{X} \to \mathbb{R}$ to denote the random prior function implied by our model. We evaluate realisations of this random function by multiplying weight vectors drawn from the prior over weights with the basis expanded inputs as

$$f(\cdot) = \phi(\cdot)w \quad \text{with} \quad w \sim \mathcal{N}(0, A^{-1}), \tag{2.5}$$

and display them in Figure 2.1. We denote by X the array of inputs $(\phi(x_i))_{i=1}^n$, and with $f(X) \in \mathbb{R}^n$ the vector given by our prior random function evaluated at these inputs. Pushing the prior distribution over weights through the product with the feature expansion, we obtain the prior distribution over function values evaluated at the inputs

$$f(X) \sim \mathcal{N}(0, \Phi A^{-1} \Phi^T). \tag{2.6}$$

We visualise the covariance matrices for our affine and random Fourier basis in Figure 2.2.

The choice of basis affects our model's uncertainty a priori and thus how much data will be needed to pin down accurate values for the parameters. If we choose a more flexible function class, then we will need more data to constrain the parameters and vice versa. The Fourier model with a large value for ψ is more flexible than the affine model since it can express non-linear functions in the inputs. This additional flexibility is reflected in the covariance matrix structures shown in Figure 2.2. The linear model assumes strong correlations throughout the input space. Only a few observations will be enough to constrain its parameters everywhere. On the other hand, the Fourier model's band diagonal covariance structure tells us the model assumes that targets are only correlated when their inputs are nearby. How close the inputs need to be is given by the width of the diagonal band. Since each observation will only constrain the random functions locally, many more observations are necessary to reduce the Fourier model's uncertainty. Since there are more ways for a

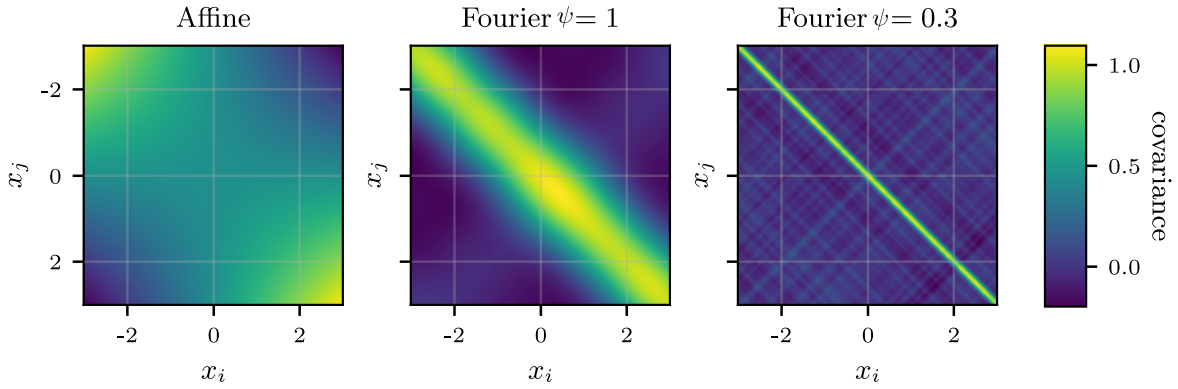


Fig. 2.2 Covariance matrices of the prior distribution over functions evaluated at 501 equally spaced points in the range $[-3, 3]$ The left side plot uses an affine basis expansion (2.3), the middle one a 500 element random Fourier expansion with a Gaussian spectral measure and a lengthscale of $\psi = 1$ (2.4), and the right side plot uses a similar Fourier expansion but with a lengthscale of $\psi = 0.3$.

function to change quickly than slowly, a smaller value of ψ leads to an even more flexible random Fourier model with a thinner band-diagonal covariance structure. This model will require even more data to learn.

Suitable feature expansions exist for many types of data, such as images (van der Wilk et al., 2017), natural text (Collins and Duffy, 2001), and even graphs (Tripp et al., 2023). Throughout this chapter we will use the Fourier basis as a recurring example. As we will see in Section 2.2.2, the random Fourier linear model is intimately tied to stationary Gaussian processes.

2.1.2 Posterior inference: from loss functions to distributions

Having discussed the choice of model, we now turn to learning from data. Intuitively, learning can be thought of as combining what we knew a priori with the information that the newly observed data tells us. We can achieve this by scoring candidate parameter vectors by their density under our prior Π and how closely the corresponding functions pass to the observed targets (the mapping between weight vectors and functions is given in (2.5)). The latter requirement is quantified by the probability density of our observations given the weights, which is known as the likelihood when taken as a function of the weights. Our assumption on the Gaussianity of the observation noise implies the conditional density over the targets

is $p(Y|w) = \mathcal{N}(Y; \Phi w,\, B^{-1})$. We assume iid inputs, making this density factorise across observations as $\prod_{i=1}^{n} \mathcal{N}(y_i; \phi(x_i)w,\, B_i^{-1})$.

Since we require our functions to simultaneously be constrained by our prior *and* likelihood, we construct an objective function by multiplying the two, obtaining the joint density $p(Y|w)\pi(w) = \prod_{i=1}^{n} p(y_i|w)\pi(w)$. Taking a logarithm[2] for numerical stability, we find that the likelihood corresponds to the least squares regression loss and the prior, to the sum of squares regulariser, both up to an additive constant. That is, we obtain the loss $\mathcal{L} : \mathcal{R}^d \to R_+$ given by

$$\log p(Y|w) + \log \pi(w) + C = \underbrace{\frac{1}{2}\sum_{i=1}^{n} \|y_i - \phi(x_i)w\|^2_{B_i}}_{\text{least squares loss}} + \underbrace{\frac{1}{2}\|w\|^2_A}_{\text{regulariser}} \coloneqq \mathcal{L}(w), \tag{2.7}$$

where C is the additive constant independent of w and Y. Both terms in the expression are quadratic, with the curvature of the fit term being $M = \Phi^T B \Phi^T$ and the regulariser's curvature being given by A. The curvature of the full loss is thus $\nabla^2_w \mathcal{L} = M + A \coloneqq H$. This allows for a closed form solution for the maximum a posteriori (MAP) estimate of the parameters $w_\star = H^{-1}\Phi^T BY$. Thus the MAP function is $f_\star(\cdot) = \phi(\cdot)w_\star$. We refer to Bishop (2006) for more detailed derivations.

Only finding the optima of the loss does not tell us how confident we should be in the corresponding parameter setting. For instance, if there are many parameter settings obtaining similar loss values but mapping to very different functions, i.e. the determinant of H is small, we might become less confident in the MAP estimate. To fully capture the uncertainty in our parameter estimate we resort to Bayesian inference. We obtain the posterior density over parameters through Bayes rule

$$\pi(w|Y) = \frac{p(Y|w)\pi(w)}{\int_w p(Y|w)\, d\pi(w)}. \tag{2.8}$$

From (2.7) it is clear that the posterior relates to the linear regression loss as $\pi(w|Y) \propto \exp(-\mathcal{L}(w))$[3]. Since the loss is quadratic, the posterior is also Gaussian with mean $w_\star$ and covariance $H^{-1} \coloneqq \Sigma$. We illustrate this for our affine model in Figure 2.3. The ratio of the joint density to the posterior $p(Y) = \int_w p(Y|w)\, d\pi(w)$ is known as the "evidence", a constant independent of w, which we will discuss in detail in Section 2.4.

[2]The monotonicity of the logarithm ensures the optima of the function do not change

[3]It is worth noting that we can use this strategy to construct probability distributions from other positive-valued functions.

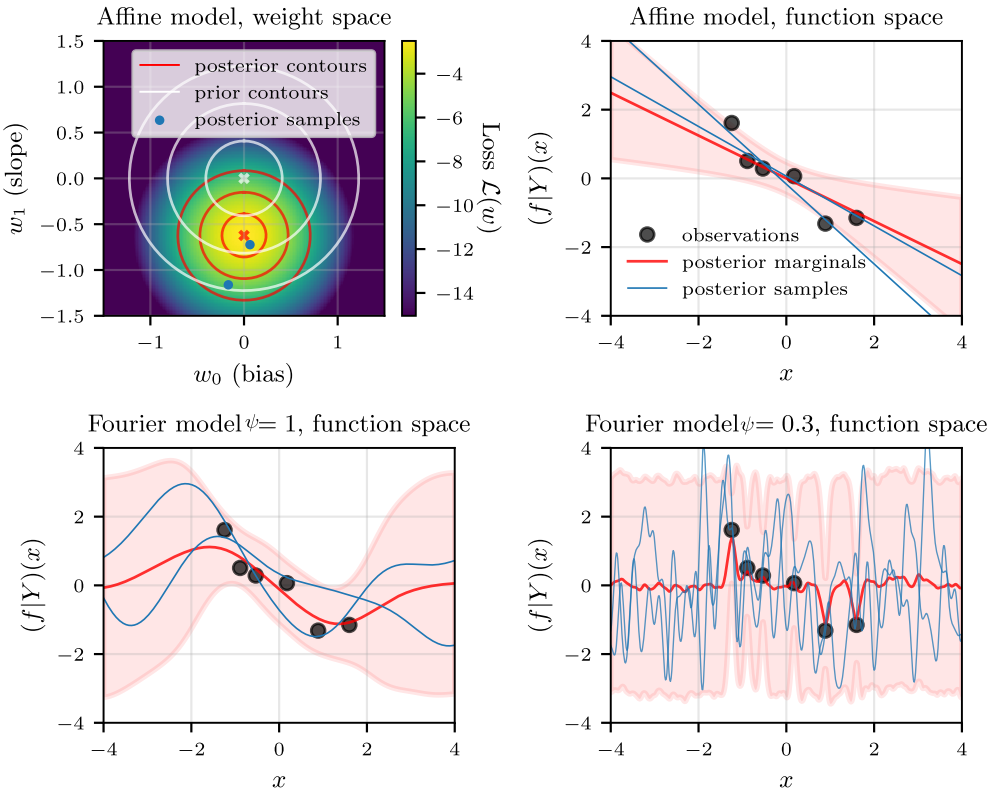


Fig. 2.3 The top left plot shows the $d = 2$ dimensional posterior landscape of our affine model fit on a $n = 6$ observation dataset with $B = 2I$ and $A = 6I$. The 1, 2 and 3 standard deviation prior and posterior contours are overlayed on top. We draw 2 samples from the weight space posterior, which we plot as function samples in the top right plot. The top right plot also displays the mean and 2 standard deviation contours of the posterior random function $f|Y$. The bottom left and bottom right plots display the same objects as the top right, but for the 500 element random Fourier basis with a Gaussian spectral measure. We set $A = 0.4I$ for the Fourier models. The lengthscale on the left is $\psi = 1$ and the right side plot uses $\psi = 0.3$.

We draw from the posterior distribution over functions by multiplying posterior weight samples with our basis expansion

$$\begin{aligned}(f|Y)(\cdot) = \phi(\cdot)w \quad \text{with} \quad w \sim \mathcal{N}(w_\star, \Sigma)\\ \Sigma = H^{-1} = (A + M)^{-1} \quad \text{and} \quad w_\star = \Sigma\Phi^T BY,\end{aligned} \tag{2.9}$$

and illustrate this for the different priors introduced in Section 2.1.1, and a small dataset, in Figure 2.3. Computationally, the cost of evaluating this posterior is dominated by computing the inverse of the Hessian H which presents cubic cost in the number of observations times output dimensions $\mathcal{O}\left((nc)^3\right)$.

At an array of test inputs $X' = (x'_i)_{i=1}^{n'}$ with corresponding featurisation $\Phi' \in \mathbb{R}^{n'c \times d}$, we evaluate the posterior distribution over function values by marginalising out the parameters in (2.9). Since we are dealing with a linear combination of Gaussian variables, the distribution over function evaluations will be jointly Gaussian

$$(f|Y)(X') \sim \mathcal{N}(\Phi' w_\star,\, \Phi' \Sigma \Phi'^T). \tag{2.10}$$

We illustrate the marginals of this distribution in Figure 2.3. The affine model presents the smallest posterior errorbars, as it is the least flexible. We can pin down the value of its parameters with the least amount of data. The $\psi = 1$ Fourier model presents a smoother posterior mean and larger errorbars, a consequence of the model's increased flexibility. Additionally, the band diagonal structure of the Fourier model's covariance (recall Figure 2.2) results in the posterior returning to the prior covariance far enough away from our observations. For the model with lengthscale $\psi = 0.3$, this happens so fast that the posterior ends up matching the 0-mean prior almost everywhere, except very close to the data. Visual inspection reveals this model choice is too flexible for our toy dataset. We would not expect this solution to generalise to additional observations. This lack of generalisation is also reflected in the large errorbars of the posterior.

So far, we have looked at the posterior distribution over functions. However, if we want to make predictions about observations, we need to take into account that these are generated as noisy function realisations. The output space distribution that accounts for both the uncertainty in our parameters and observations is the posterior predictive. For a new input x_{n+1}, the posterior predictive density over the corresponding target y_{n+1} is given by

$$p(y_{n+1}|Y) = \int_w p(y_{n+1}|w)\, d\pi(w|Y). \tag{2.11}$$

In the linear-Gaussian setting, assuming a homoscedastic observation noise of precision b, this density corresponds to the distribution $\mathcal{N}(\phi(x')w_\star,\, \phi(x')\Sigma\phi(x')^T + b^{-1}I)$.

2.2 The function space view: Gaussian processes

A stochastic process is a potentially infinite set of random random variables. We say that a random function $f : \mathcal{X} \to \mathbb{R}^c$ is a *Gaussian process* if, for every finite set of points $X \in \mathcal{X}^n$, $f(X)$ is jointly Gaussian. Both of the expressions we derived in the previous section for the prior (2.6) and posterior (2.10) distributions over function evaluations are multivariate

Gaussians, satisfying this definition. Viewing the Gaussian linear model as a Gaussian process (GP) will allow us to perform Bayesian inference without ever having to work with the parameters w directly. The use of stochastic processes as priors is known as Bayesian nonparametrics (Ghosal and van der Vaart, 2017).

2.2.1 Duality

Instead of the usual measure theoretic definition of stochastic processes (Matthews, 2017), we will derive the function-space view as the convex dual formulation of the Gaussian linear model (Khan, 2014). We will make heavy use of this duality throughout the rest of the thesis.

Derivation Convex duality of Gaussian linear regression loss
We begin by formulating the linear regression loss $\mathcal{L}$ introduced in (2.7) as the constrained optimisation problem

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}\sum_{i=1}^{n} \|y_i - u_i\|^2_{B_i} + \frac{1}{2}\|w\|^2_A$$
$$\text{s.t.} \quad u_i = \phi(x_i)w \quad \forall i. \tag{2.12}$$

We introduce the Lagrangian $L : \mathbb{R}^d \times \mathbb{R}^{nc} \times \mathbb{R}^{nc} \to \mathbb{R}_+$ with Lagrange multiplier $\alpha \in \mathbb{R}^{nc}$ as

$$L(w, u, \alpha) = \frac{1}{2}\sum_{i=1}^{n} \|y_i - u_i\|^2_{B_i} + \frac{1}{2}\|w\|^2_A + \sum_{i=1}^{n}\langle \alpha_i, u_i - \phi(x_i)w\rangle. \tag{2.13}$$

This problem is quadratic in both w and u and Slater's condition holds (see 5.2 in Boyd and Vandenberghe (2014)). Thus there is strong duality

$$\min_{w\in\mathbb{R}^d} \mathcal{L}(w) = \max_{\alpha\in\mathbb{R}^{nc}} \inf_{w\in\mathbb{R}^d} \inf_{u\in\mathbb{R}^{nc}} L(w, u, \alpha) = \max_{\alpha\in\mathbb{R}^{nc}} L(w', u', \alpha) \tag{2.14}$$

where the optimal primal variables can be shown, by solving the respective quadratic problems, to be given by $w' = A^{-1}\Phi\alpha, \quad \text{and} \quad u'_i = y_i - B_i^{-1}\alpha_i$. Plugging these in to the Lagrangian yields the dual loss

$$L(w', u', \alpha) = -\frac{1}{2}\|\alpha\|^2_{(B^{-1}+\Phi A^{-1}\Phi^T)} + \alpha^T Y \tag{2.15}$$

which is also quadratic but with curvature $B^{-1} + \Phi A^{-1}\Phi^T$. It is optimised by taking

$$\alpha_\star \coloneqq (B^{-1} + \Phi A^{-1}\Phi^T)^{-1}Y \tag{2.16}$$

Thus we can reparametrise the maximum a posteriori function estimate in terms of the optimal Lagrange multipliers $\alpha_\star$ as

$$f_\star(\cdot) = \phi(\cdot)w_\star = \phi(\cdot)A^{-1}\Phi^T\alpha_\star. \tag{2.17}$$

There are nc Lagrange multipliers in the vector $\alpha_\star$, one per observation and output dimension. Obtaining them requires solving a nc dimensional system at cost $\mathcal{O}((nc)^3)$. This is in contrast to the $\mathcal{O}(d^3)$ cost of the primal solution $w_\star$. Thus, the dual formulation will be preferable when $nc < d$.

An analogous derivation to the one above, given in (Khan, 2014), can be used to find the dual formulation of the full Gaussian posterior, including the covariance. However, a faster route is to use the Woodbury matrix to re-write the expression for the posterior covariance into a form that depends on $(B^{-1} + \Phi A^{-1}\Phi^T)^{-1}$, the curvature of the dual problem, as

$$\Sigma = (A + \Phi^T B\Phi)^{-1} = A^{-1} - A^{-1}\Phi^T(B^{-1} + \Phi A^{-1}\Phi^T)^{-1}\Phi A^{-1}. \tag{2.18}$$

With this, the posterior distribution over functions evaluated at a set of test points $X' = (x'_i)_{i=1}^{n'}$ with featurisation $\Phi' \in \mathbb{R}^{n'c\times d}$ can be written as

$$\begin{aligned}(f|Y)(X') \sim \mathcal{N}(&\Phi' A^{-1}\Phi^T(B^{-1} + \Phi A^{-1}\Phi^T)^{-1}Y,\\ &\Phi' A^{-1}\Phi'^T - \Phi' A^{-1}\Phi^T(B^{-1} + \Phi A^{-1}\Phi^T)^{-1}\Phi A^{-1}\Phi'^T).\end{aligned} \tag{2.19}$$

Again, evaluating this expression presents cost $\mathcal{O}\left((nc)^3\right)$ as opposed to $\mathcal{O}(d^3)$ for the primal form (2.10).

2.2.2 From features to kernels

When working with the dual form of the Gaussian linear model, we no longer encounter the featurised design matrix $\Phi \in \mathbb{R}^{nc\times d}$ explicitly; it only shows up as part of the $nc \times nc$ matrix $\Phi A^{-1}\Phi^T \coloneqq K$, which we will refer to as the *kernel matrix*.

Taking $c = 1$ for simplicity of notation but without loss of generality, any feature map of the form $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$ defines a symmetric and positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

as $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ for $x_i, x_j \in \mathcal{X}$. The converse is also true; any symmetric and positive definite kernel k can be written as an inner product in some RKHS $\mathcal{H}$ (Aronszajn, 1950). We now note that the positive definite matrix A can be absorbed into the featurised design matrices. It can simply be seen as a rotation and shear of the features. Thus, there exists a kernel that generates our kernel matrix such that $[K]_{i,j} = k(x_i, x_j) \, \forall \; i, j = 1, .., N$. This fact will allow us to avoid working with features entirely in favour of their inner products. In turn, this will allow us to use potentially infinite dimensional feature expansions, where it may be impossible to explicitly compute the features. The substitution of input inner products $\langle x_i, x_j \rangle$ with kernel function evaluations $k(x_i, x_j)$ to obtain a non-linear (in the inputs) version of existing algorithms is known as the *kernel trick* (Scholkopf and Smola, 2001). Additionally, we will henceforth denote the matrix built by evaluating our kernel at all pairs in two arrays of inputs X and X' as $K_{XX'}$. That is $[K_{XX'}]_{i,j} = k(x_i, x'_j) : i = 1, 2, \ldots, n, \; j = 1, 2, \ldots, n'$. We refer to Hofmann et al. (2006) for a tutorial on RKHS.

To illustrate the kernel trick, we consider the random Fourier basis given in (2.4) and let the number of features d go to infinity. We recover the squared exponential or Radial Basis Function (RBF) kernel

$$\begin{aligned} k(x_i, x_j) &= \langle \phi_{s,r}(x_i), \phi_{s,r}(x_j) \rangle \\ &= \frac{2}{d} \sum_{l=1}^{d} \cos(s_l^T x_i + r_l) \cos(s_l^T x_j + r_l) \underset{d=\infty}{\rightarrow} \exp\left(\frac{-\|x_i - x_j\|^2}{\psi^2} \right) \\ \text{with} \quad s_l &\sim \mathcal{N}(0, \psi^{-2}) \qquad \text{and} \qquad r_l \sim \text{Uniform}(0, 2\pi). \end{aligned} \tag{2.20}$$

Thus, when we use the RBF kernel we are leveraging an infinite dimensional feature expansion without ever having to compute Fourier features explicitly. We will discuss random feature approximations to kernels in more detail in Section 2.2.4.

We refer to the partial evaluation of the kernel $k(x, \cdot) = \phi(x) A^{-1} \phi(\cdot)^T : \mathcal{H} \rightarrow \mathbb{R}$ as the *evaluation functional*, which is an element of RKHS in its own right[4]. Its name comes from the fact that for a kernel k, there is a unique $k(x, \cdot) \in \mathcal{H}$ which evaluates a function $\varphi \in \mathcal{H}$ at the input $x \in \mathcal{X}$ through the inner product

$$\varphi(x) = \langle \varphi, k(x, \cdot) \rangle = \sum_{i=1}^{d} \alpha_i k(x_i, x). \tag{2.21}$$

[4]To see this note that elements of the RKHS can be written as $\sum_{i=1}^{d} \alpha_i k(x_i, \cdot)$ and then choose all but 1 α_i to be 0.

This is the reproducing property, which gives name to the RKHS. A consequence of this property is that $\langle k(x,\cdot), k(x',\cdot)\rangle = k(x_i, x_j)$.

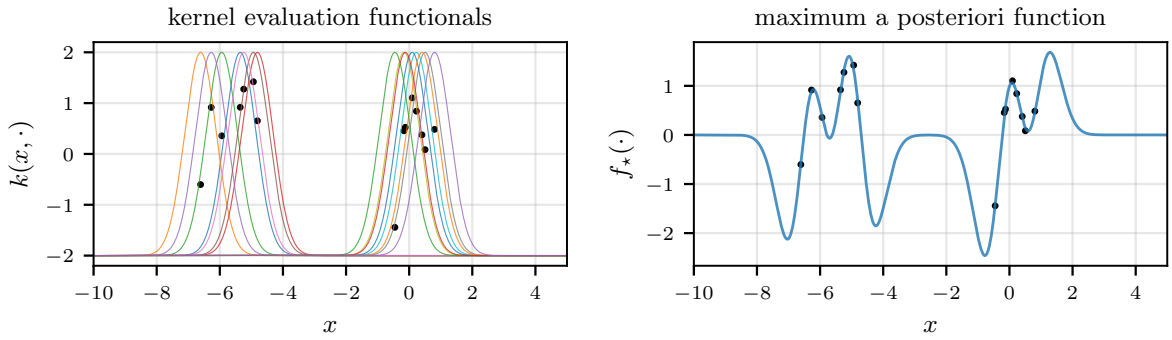


Fig. 2.4 Left: RBF kernel ($\psi = 0.5$) evaluation functionals for each observation (black dots) in a toy 1d dataset. Right: the posterior mean function is a linear combination of evaluation functionals.

We can now identify the dual expression for the posterior mean function $f_\star(\cdot) = \phi(\cdot)A^{-1}\Phi^T\alpha_\star$, given in (2.17), as a linear combination of evaluation functionals

$$f_\star(\cdot) = \sum_{i=1}^{n} \alpha_{\star,i} k(\cdot, x_i) = K_{(\cdot)X}\alpha_\star, \tag{2.22}$$

where for the last equality we write $K_{(\cdot)X}$ for the stacked evaluation functionals at the observed datapoints $k(\cdot, x_1), \ldots, k(\cdot, x_n)$, allowing us express functions in $\mathcal{H}$ as matrix vector products. We can think of the evaluation functionals as a basis function expansion of the inputs x_i, $i < n$. The entries of the linear coefficient vector $\alpha_\star$ are known as the *representer weights*[5]. Figure 2.4 depicts a set of evaluation functionals for the RBF kernel (2.20) and how the posterior mean function is constructed as a linear combination of these functions. The local nature of the kernel leads to the evaluation functionals going to 0 far enough away from the observations and this behaviour translates to the MAP function $f_\star$.

2.2.3 Bayesian reasoning about functions: Gaussian processes

We now leverage duality and the kernel trick to re-state the Bayesian model from Section 2.1 directly as a Gaussian process

$$Y = f(X) + \mathcal{E} \quad \text{with} \quad f \sim \text{GP}(\mu, k) \quad \text{and} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}). \tag{2.23}$$

[5]This name is due to the representer theorem of Scholkopf and Smola (2001).

The *mean function* $\mu(\cdot) = \mathbb{E}(f(\cdot))$ and a *covariance kernel* $k(\cdot, \cdot') = \text{cov}(f(\cdot), f(\cdot'))$ uniquely identify the Gaussian process prior. Without loss of generality, we will assume $\mu(\cdot) = 0$ throughout the rest of this chapter.

The posterior distribution over functions is another Gaussian process $f|Y \sim \text{GP}(f_\star, k_\star)$ with

$$\begin{aligned} f_\star(\cdot) &= K_{(\cdot)X}(K + B^{-1})^{-1}Y \\ k_\star(\cdot, \cdot') &= k(\cdot, \cdot') - K_{(\cdot)X}(K + B^{-1})^{-1}K_{X(\cdot')}. \end{aligned} \tag{2.24}$$

Evaluating both of these expressions present a cost cubic in the number of observations and output dimensions $\mathcal{O}\left((nc)^3\right)$.

2.2.4 Sampling from Gaussian processes & random features

We saw in (2.5) how to sample from the prior distribution over functions by first sampling the weights from the prior $w \sim \mathcal{N}(0, A^{-1})$ and taking an inner product with the feature expansion $\langle \phi(\cdot), w \rangle$. This operation presents a linear cost in the number of features d, resulting computationally intractable when dealing with an infinite dimensional feature space, such as the one associated with the RBF kernel (2.20).

Matrix square root sampling Instead, from (2.6), we know that the distribution over prior function samples evaluated at a pre-fixed set of points $X' \in \mathcal{X}^{n'}$ is $\mathcal{N}(0, K_{X'X'})$. Thus, we can evaluate a prior sample at X' by transforming an n' dimensional vector of standard Gaussian noise with a matrix square root of the covariance. For instance, we may use the Cholesky decomposition $\text{L}\text{L}^T = K_{X'X'}$ to compute

$$f(X') = \text{L}u \quad \text{with} \quad u \sim \mathcal{N}(0, I_{cn'}). \tag{2.25}$$

Be that as it may, this approach requires knowing the points at which we want to evaluate our prior functions a priori and presents a cost cubic in the number of points we want to evaluate at $\mathcal{O}\left((n'c)^3\right)$. Furthermore, if X' contains repeated points or pairs of points for which the kernel evaluates to very small values, $K_{X'X'}$ may be singular or close to singular, resulting in numerical instability when computing its square root.

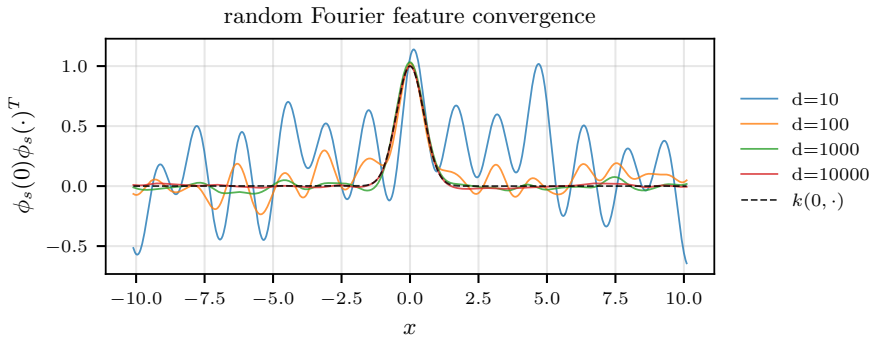


Fig. 2.5 Convergence of random Fourier feature basis (given in (2.4)) to the RBF kernel's evaluation functional $k(0,\cdot)$ using the estimator in (2.26) as the number of random features d increases.

Random feature prior sampling Fortunately, we may approximate prior function samples to high accuracy using random features (Rahimi and Recht, 2007; Terenin, 2022; Wilson et al., 2020). In particular, we may use some feature expansion $\phi_s : \mathcal{X} \to R^{c\times d}$ parametrised by a random variable s with law Ω to construct an unbiased estimator of a kernel function as

$$k(x,x') = \mathbb{E}_{s\sim\Omega}\,\phi_s(x)\phi_s(x')^T. \tag{2.26}$$

We can use these random features to construct a Monte Carlo estimator of a prior function sample $f \sim \text{GP}(\mu,k)$ as

$$f(\cdot) \approx \widetilde{f}(\cdot) = \phi_s(\cdot)w \quad \text{with} \quad w \sim \mathcal{N}(0,I_d) \quad \text{and} \quad s \sim \Omega \tag{2.27}$$

at $\mathcal{O}(d)$ cost, where d is the dimensionality of the feature expansion, often referred to as the number of random features. This parameter controls the error in the approximation, which goes to 0 as d goes to infinity. We have approximately reversed the kernel trick, recovering a finite dimensional linear model. We may now evaluate our prior function sample at any $x \in \mathcal{X}$ by simply evaluating the random features at x and taking an inner product with the random weights. Following Wilson et al. (2020), both the next section and Chapter 4 will efficiently draw approximate posterior function samples by replacing instances of f with $\widetilde{f}$.

Random Fourier features (2.4) can be used to approximate any stationary kernel—that is, those that can be written as $k(x,x') = k'(x-x')$ for $k' : \mathcal{X} \to \mathbb{R}$—by taking the distribution from which the cosine frequencies are sampled Ω to be the normalised spectral measure of the kernel k. As we saw in (2.20), the RBF kernel is recovered when Ω is chosen to be Gaussian. We illustrate the convergence of this estimator in Figure 2.5. More sophisticated Fourier feature sampling strategies have been developed to reduce the variance of the above

estimators (Reid et al., 2023; Yu et al., 2016). Some non-stationary kernels also admit random features. For instance, there exist random features that describe graphs Reid et al. (2024), ones that describe sets of binary attributes (Tripp et al., 2023), and ones that approximate the attention mechanism (Peng et al., 2021).

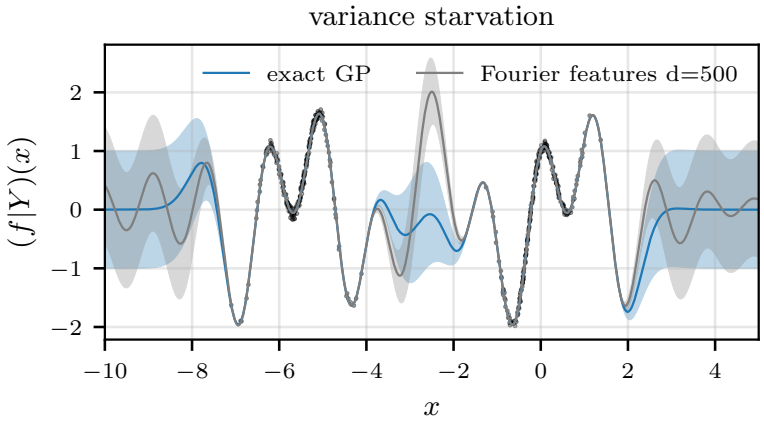


Fig. 2.6 Illustration of variance starvation when using random Fourier features to approximate a posterior GP. The shaded region represents a one standard deviation credible interval. Our $n = 5000$ datapoints are placed close to each other and are largely redundant. This consumes the degrees of freedom of our $d = 500$ random features, leading to arbitrary extrapolation and reduced uncertainty away from the training data. This doesn't happen with our exact GP, which effectively uses an infinite number of basis functions.

Variance Starvation Random Fourier features can also be used for approximate posterior inference at cost $\mathcal{O}(d^3)$. For this, we simply approximate the infinite feature expansion with d random features and then proceed with linear model inference as in (2.9). However, this is not advisable, as the degrees of freedom needed to represent posterior functions grow with the number of observations and $d \gg n$ features are often needed to obtain a good approximation. This issue is known as variance starvation. It is intimately related to Gibbs ringing and is discussed in detail in 2.4.2 of Terenin (2022). We illustrate variance starvation in Figure 2.6.

2.3 Pathwise view: working with the posterior random function

We have so far characterized inference in the Gaussian linear model in terms of the posterior distribution over its weights and the posterior Gaussian process. These require dealing with the posterior covariance matrices over weights and observations, respectively. Even storing

these in memory can be computationally intractable when the number of parameters or observations is large, which is the setting of interest of this thesis. This section introduces *pathwise conditioning* (Wilson et al., 2020, 2021), a formulation of inference that deals only with posterior weight or function samples, as opposed to complete posterior distributions. In turn, this will allow us to deal with vectors of dimension d or nc, as opposed to covariance matrices, which are quadratic in those quantities. We will build upon the pathwise view of inference to design scalable approximate inference algorithms in Chapter 4.

For the Bayesian model in (2.23), one can write the posterior random function directly as

$$\begin{aligned}(f|Y)(\cdot) = f(\cdot) + K_{(\cdot)X}(K + B^{-1})^{-1}(Y - f(X) - \mathcal{E}) \quad \text{with} \\ \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad f \sim \text{GP}(\mu, k).\end{aligned} \tag{2.28}$$

It is straight forward to check that the moments of $(f|Y)$ match those of the posterior GP given in (2.24). Thus, evaluating this expression for a particular prior function sample f and noise sample $\mathcal{E}$ yields a posterior function sample. Although we retain the cubic cost of a linear solve against $(K + B^{-1})$, this only needs to be done once. Then we are free to evaluate the posterior sample at any set of test points X' at only linear cost in nc. Additionally, we avoid the need to store the covariance matrix explicitly.

To gain a better understanding of the pathwise form of the posterior, we can rewrite it as a sum of three terms

$$\begin{aligned}(f|Y)(\cdot) = \underbrace{f_\star(\cdot)}_{\text{posterior mean}} + \underbrace{f(\cdot)}_{\text{prior sample}} - \underbrace{K_{(\cdot)X}(K + B^{-1})^{-1}(f(X) + \mathcal{E})}_{\text{uncertainty reduction term}} \\ \text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad f \sim \text{GP}(0, k),\end{aligned} \tag{2.29}$$

which are illustrated in Figure 2.7. The first component is the posterior mean function $f_\star(\cdot) = K_{(\cdot)X}\alpha_\star$, which we analysed in Section 2.2.2. Its job is ensuring our posterior function samples pass near the datapoints. To it, we add a prior function sample, whose value will vary across input space in a data-independent way. The uncertainty reduction term cancels the effect of the prior function sample near the datapoints.[6] It ensures the posterior function sample takes values close to the posterior mean, and thus close to our observed targets, near the training data. Just like the posterior, the uncertainty reduction term takes the form of a linear combination of evaluation functionals $K_{(\cdot)X}\alpha_u$, with $\alpha_u = (K + B^{-1})^{-1}(f(X) + \mathcal{E})$. Consequently, far away from the observed data, the posterior function samples revert to the prior function samples, inflating the uncertainty in the posterior to match the prior uncertainty.

[6]We say that two points x_i and x_j are "near" when $k(x_i, x_j)$ is small.

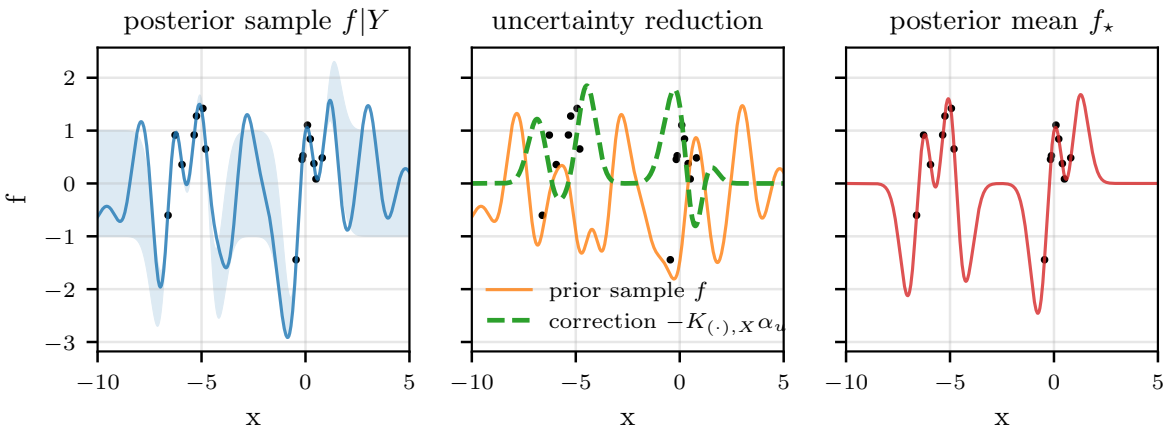


Fig. 2.7 Illustration of the pathwise construction of the posterior function sample, shown on the left together with a single standard deviation posterior credible region contour. The middle plot shows a prior function sample together with its corresponding uncertainty reduction term, which cancels the prior sample near the training data. The right side plot shows the GP posterior mean function, which is added to the prior sample and uncertainty reduction term to build the posterior sample.

The pathwise formulation first appeared in the field of geostatistics, where it was referred to as "Matheron's rule" (Journel and Huijbregts, 1978). It has been used to perform inferences in astrophysics models (Hoffman, 2009; Hoffman and Ribak, 1991) and Gaussian Markov random fields (Papandreou and Yuille, 2010). More recently, it was re-discovered and popularised among the Gaussian process community by Wilson et al. (2020), to whom the form (2.28) is due.

Remark Are GP function samples in the RKHS?
The pathwise formulation of GP posterior samples (2.29) allows us to answer this question; we must simply check the norm of each term in the RKHS. For the posterior mean, we have

$$\|f_\star\|_{\mathcal{H}}^2 = \left\langle K_{(\cdot)X}\alpha_\star, K_{(\cdot)X}\alpha_\star \right\rangle = \alpha_\star^T K \alpha_\star$$

which will be a finite number as long as the number of observations times output dimensions nc is finite. We can use the same argument for the uncertainty reduction term $K_{(\cdot)X}\alpha_U$. However, this is not necessarily true for the prior sample. For infinite dimensional feature expansions, it can not be written as a linear combination of a finite number of basis functions. Its RKHS norm may be infinite $\|f\|_{\mathcal{H}} = \infty$. Thus, neither the GP prior or posterior functions live in the RKHS associated with the GP's

covariance kernel k. However, the difference between the GP prior and posterior functions always lives in the RKHS $f - (f|Y) \in \mathcal{H}$.

2.3.1 Efficiently sampling from GP posteriors with random features

The practical utility of the pathwise formulation (2.28) rests on our ability to efficiently evaluate a prior function sample. In the infinite-dimensional feature case, this can present a number of challenges, discussed in Section 2.2.4. However, following Wilson et al. (2020), we can efficiently approximate the pathwise form of posterior functions using a random feature approximation of the prior

$$\begin{aligned}(f|Y)(\cdot) \approx \widetilde{f}(\cdot) + K_{(\cdot)X}(K + B^{-1})^{-1}(Y - \widetilde{f}(X) - \mathcal{E}) \quad \text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \\ \text{and} \quad \widetilde{f}(\cdot) = \phi_s(\cdot)w \quad w \sim \mathcal{N}(0, I_d) \quad s \sim \Omega. \end{aligned} \tag{2.30}$$

Importantly, random features are only used to approximate the prior function sample. Conditioning on the data is done via the exact linear solve, at cubic cost in the number of observations and output dimensions, avoiding variance starvation. Pathwise sampling combined with random features provides a very powerful toolkit for decision making under uncertainty which we will use throughout this thesis.

2.3.2 Duality between pathwise conditioning and sample-then-optimise

We now present the primal form of the pathwise formulation of posterior samples for finite dimensional feature spaces and show that it is equivalent to the "sample-then-optimise" posterior sampling strategy for Bayesian linear models (de G. Matthews et al., 2017)[7]. For this, we start from the pathwise expression of the posterior over weights

$$\begin{aligned} w|Y = w_0 + A^{-1}\Phi^T(\mathrm{B}^{-1} + K)^{-1}(\mathcal{E} - \Phi w_0) \\ \text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}), \end{aligned} \tag{2.31}$$

which matches (2.28) but we have removed the product with the feature expansion that maps weight samples to function samples, that is $f(\cdot) = \phi(\cdot)w$. Despite returning a d dimensional weight sample, (2.31) retains a linear solve against $(\mathrm{B}^{-1} + K)$ at cost $\mathcal{O}\left((nc)^3\right)$.

[7]Although, I believe this observation to first have been made in Antorán et al. (2023), which forms the basis of Chapter 6, it is presented here as it constitutes a useful building block for the rest of the thesis.

Derivation Duality of pathwise conditioning and sample-then-optimise
We recall that $H = A + \Phi^T \mathrm{B} \Phi$ and then apply the following series of matrix identities to (2.31)

$$
\begin{aligned}
w|Y &= w_0 + A^{-1}\Phi^T(\mathrm{B}^{-1} + \Phi A^{-1}\Phi^T)^{-1}(\mathcal{E} + Y - \Phi w_0) && (2.32)\\
&= w_0 + A^{-1}\Phi^T\mathrm{B}(I + \Phi A^{-1}\Phi^T\mathrm{B})^{-1}(\mathcal{E} + Y - \Phi w_0) && (2.33)\\
&= w_0 + A^{-1}(I + \Phi^T\mathrm{B}\Phi A^{-1})^{-1}\Phi^T\mathrm{B}(\mathcal{E} + Y - \Phi w_0) && (2.34)\\
&= w_0 + H^{-1}\Phi^T\mathrm{B}(\mathcal{E} - \Phi w_0) && (2.35)\\
&= H^{-1}((H - \Phi^T\mathrm{B}\Phi)w_0 + \Phi^T\mathrm{B}(\mathcal{E} + Y)) && (2.36)\\
&= H^{-1}(\Phi^T\mathrm{B}(\mathcal{E} + Y) + Aw_0) && (2.37)\\
&\text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}).
\end{aligned}
$$

Equation (2.37) recovers an expression containing a linear solve against H, with cost $\mathcal{O}(d^3)$. By visual inspection, we can identify that (2.37) matches the form of the maximum a posteriori weight setting for weight-space model (2.9), but where our targets are perturbed by adding $\mathcal{E}$ and our prior mean is w_0. Thus, (2.37) represents the solution to a quadratic problem analogous to the linear regression loss

$$
\begin{aligned}
w|Y &= \min_{w \in \mathbb{R}^d} \frac{1}{2}\|Y + \mathcal{E} - \Phi w\|^2_{\mathrm{B}} + \frac{1}{2}\|w - w_0\|^2_A && (2.38)\\
&\text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}).
\end{aligned}
$$

This expression is known in the literature as the "sample-then-optimise" objective. de G. Matthews et al. (2017) use it in the noiseless case ($B^{-1} = 0$) to study the connection between Bayesian inference and optimisation in overparametrised linear models. Osband et al. (2018) and Pearce et al. (2020) apply variants of the objective to draw approximate posterior samples from the posterior distribution over the weights of a neural network, although this approximation may be very crude.

Leveraging the pathwise or sample-then-optimise formulations for posterior sampling at scale (large d and large nc) still requires solving large linear systems, which remains an open problem. Standard methods, such as conjugate gradients and matrix sketching are discussed in Chapter 3. In Chapter 4 we instead propose to use stochastic gradient descent for pathwise inference.

2.3.3 Demonstration: pathwise conditioning for Bayesian optimisation

We conclude by reviewing how we may use pathwise inference for efficient decision making. In particular, consider the problem of finding the input which maximises some unknown function $g : \mathcal{X} \to \mathbb{R}$ in the least number of function evaluations. To this end, we place a GP prior over the function and choose new points at which to evaluate g as

$$x_{\text{new}} = \arg\max_{x' \in \mathcal{X}} \int \mathcal{U}(x', f)\, dP_{f|Y}, \tag{2.39}$$

where $P_{f|Y}$ is the measure of the posterior GP and $\mathcal{U} : \mathcal{X} \times \mathbb{R}^{\mathcal{X}} \to R$ is a utility function (Hansson, 2011). The latter is chosen to trade-off exploration and exploitation.

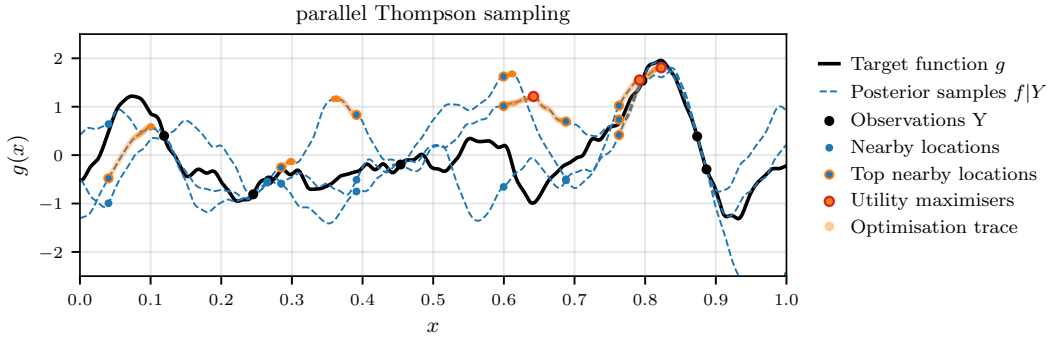


Fig. 2.8 Illustration of parallel Thompson sampling procedure with multistart gradient-based optimisation of posterior function samples. Our GP is initialised with 7 observations from g corresponding to inputs chosen uniformly at random from $[0, 1]$.

For our example, we will use Thompson sampling (Thompson, 1933), where $\mathcal{U}(x, f) = \mathbb{1}(f(x) = \max_{x' \in \mathcal{X}} f(x'))$ where $\mathbb{1}$ is the indicator function. At each step, we approximate the integral in (2.39) with a single Monte Carlo (MC) sample. That is, we draw a single posterior function sample and choose the input that maximises it $x_{\text{new}} = \arg\max_{x' \in \mathcal{X}}(f|Y)(\cdot)$. We then evaluate $g(x_{\text{new}})$ and add it to the dataset we use to perform posterior inference in our GP.

In Figure 2.8, we demonstrate a single step of parallel Thompson sampling (Hernández-Lobato et al., 2017) in a 1d toy problem with $\mathcal{X} = [0, 1]$ and where $g \sim \text{GP}(0, k)$ with k being the Matérn $3/2$ kernel. The parallel variant differs from the above explained algorithm in that at each step we draw multiple posterior functions which we maximise to add multiple observations to our dataset at each step. We use 3 posterior functions, depicted as dashed blue lines. We add homoscedastic Gaussian noise of precision $\sqrt{1000}$ to target function evaluations. We maximise each posterior function by first evaluating it at 7 inputs chosen uniformly at random from $[0, 1]$. These are labelled "nearby locations" in the legend. We

keep the inputs corresponding to the top 3 posterior function evaluations, labelled "top nearby locations", and use the Adam optimiser to improve them until corresponding local optima of the posterior functions are found. These are labelled "utility maximisers". We evaluate the target function at the utility maximisers and add the corresponding input-observation pairs to our dataset.

The pathwise formulation of posterior functions is critical to make this algorithm computationally efficient. It allows us to solve a single linear system to obtain each posterior function per Thompson step. After this, we may evaluate each posterior function $(f|Y)$ an arbitrary number of times for its maximisation at only linear cost in the number of observations. This contrasts with the cubic cost per evaluation that we would have had to incur had we used the more-traditional definition of a posterior GP in terms of its first and second moments (2.24). The remaining bottleneck is solving the linear system to update the posterior functions when the number of observations n becomes large. This challenge will be dealt with in Chapter 4.

2.4 Model selection: the marginal likelihood, or evidence, and empirical Bayes

So far we have seen how to perform Bayesian inference over model parameters, how to transform the posterior distribution over parameters into predictions and how these predictions can be used to make decisions under uncertainty. All of these techniques rest on our prior modelling choices. In (2.2), we assumed that our targets are generated as a noisy linear combination of basis functions ϕ. Furthermore, we assume that the weights of this linear combination were sampled from a zero-centred Gaussian with precision A and that the additive observation noise is also Gaussian with precision B. We refer to these quantities (i.e. ϕ, A, B), over which we do not perform Bayesian inference, as hyperparameters and denote them by $\theta \in \Theta$. We henceforth refer to the choice of hyperparameters and the choice of model interchangeably[8]. The quality of the inferences that we do make rests on the appropriateness of our hyperparameter choices (Masegosa, 2020). Although the Bayesian framework forces us to make our modelling choices explicit, it does not tell us which choices to make.

Intuitively, we should choose our model such that it incorporates all our knowledge about the generative process of the data. As we saw in Section 2.1.1, the more restrictive the model

[8]Any model can be written in terms of a broader model class Θ which is indexed by a set of hyperparameters $\theta \in \Theta$

class, the less degrees of freedom will be left to be pinned down by the data, and the more confident we can be in our inferences. However, if our strong prior assumptions are wrong, we risk our inferences being biased and our predictions not reflecting real world outcomes.

In this section, we will depart slightly from the Bayesian framework to introduce model selection tools that efficiently navigates the bias-variance trade off. To this end, consider the integral of the likelihood against the prior, which featured as the denominator in Bayes rule (2.8). For the weight space linear model, this is

$$\log p(Y;\, \phi, B, A) = \log \mathbb{E}_{w\sim\Pi}[p(Y|w;\, \phi, B)] = \log \int_w p(Y|w;\, \phi, B)\pi(w; A)\, dw \quad (2.40)$$

which is known as the log marginal likelihood, or the model evidence. We use the semicolon ; to separate model parameters from hyperparameters on which the likelihood and the prior depend but over which we do not place a prior or perform inference. We have written out these hyperparameters in (2.40) for clarity, but we henceforth group them into the tuple $\theta = (\phi, A, B)$ for brevity. The evidence measures the degree of overlap between the prior and the likelihood, thus rewarding a choice of prior that concentrates its mass on parameter settings that fit the training data well. Too broad a prior will spread its probability mass across many models, only some of which will fit the data, decreasing the evidence. In this way, the model evidence differs from the training loss; the latter can always be improved by using a more flexible model. See chapter 28 of MacKay (2003) for additional discussion and illustrative examples.

Remark Automatic Occam's razor
In the literature, the model evidence is said to automatically incorporate "Occam's razor" since it implicitly favours "simpler" priors (Gull, 1988; Jeffreys, 1939; Mackay, 1992a; Rasmussen and Ghahramani, 2000). In this context, the notion of complexity refers to the degree of diversity of the hypotheses supported by the prior. For instance, we would say that the class of affine models is simpler than the class of third order polynomials, since the latter contains the former and many more functions. Intuitively, there are many more complex functions than simple ones.
It is important to note that "simple" does not mean more linear, continuous, or having a lower Lipschitz constant. For instance, a prior over third degree polynomials, where all of the coefficients of order greater than 0 are set to a fixed quantity—only the bias is left to be inferred—would be considered simpler and, thus preferred by the evidence, to a prior over affine models, assuming both families of functions fit the data equally well.

A complementary point of view of (2.40) is that it is the log-density of the training data when our model is set to the prior. If our prior is able to predict our training observations, then our posterior will not differ much from our prior, yielding credence to it also being able to predict yet-unseen datapoints. This intuition is formalised in the framework of PAC-Bayes bounds (Germain et al., 2016; Masegosa, 2020). Also intimately related to the model evidence are the framework of minimum description length (Grünwald, 2004) and other model selection criteria such as Akaike information criterion (Akaike, 1970) and Bayesian information criterion (Neath and Cavanaugh, 2012).

2.4.1 Comparing two models

The marginal likelihood of of some model M_1 differs from the regular likelihood in that the model parameters have been marginalised out. In this sense, it can be seen as a quantity at the second level of inference. The first level is inference over parameters, the second is over model class. We could apply this idea again to construct a third level likelihood to score members of a family of meta-model classes and so on. Thus, if we want to decide which model is best among a pair of models M_1, M_2 we can compute the ratio of their posterior probabilities at the second level of inference as

$$\frac{p(M_1|Y)}{p(M_2|Y)} = \frac{p(Y|M_1)p(M_1)}{p(Y|M_2)p(M_2)}. \tag{2.41}$$

Often the priors are chosen to be uniform over models $p(M_1) = p(M_2)$ and the posterior probability ratio matches the likelihood ratio $\frac{p(Y|M_1)}{p(Y|M_2)}$. Likelihood ratios provide a Bayesian alternative to hypothesis tests. See chapter 37 of MacKay (2003) for a detailed discussion.

Remark On the dangers of model comparison with the evidence
A criticism of marginal likelihood-based model comparison is its sensitivity to the choice of prior (Kass and Raftery, 1995). This is especially concerning when placing (seemingly) uninformative priors over our models' parameters. Intuitively, as our prior becomes fully uninformative (e.g. an improper uniform distribution over the parameters), its marginal likelihood goes to 0. In the almost-fully-uninformative regime, small changes in the prior hyperparameters, which have very little effect on posterior inferences, can have large effects on the model evidence. In response, a number of "sensitivity analysis" methods have been introduced to characterise the sensitivity of the evidence to the prior hyperparameters Sinharay and Stern (2002).

One could argue that if we are fully uncertain about the parameters of a model, and the model's predictions depend strongly on those parameters, we should be happy to throw the model in the trash. Yet, this is roughly the case with neural networks, and here we are. A different, perhaps more Bayesian view is that we should not use the evidence to perform model selection at all. Instead of discarding one model with less evidence than another, we should expand out model class and consider both models in our Bayesian model average. If we were willing to consider both models for comparison in the first place, then we must have assigned some credibility to both models a priori, and our inferences should reflect this. This is roughly the view expressed by Adrew Gelman in a blog post addressing MacKay (2003) chapter on Bayesian model comparison.[a]

[a] statmodeling.stat.columbia.edu/2011/12/04/david-mackay-and-occams-razor

2.4.2 Hyperparameter optimisation

We now extend the notion of model comparison to a continuous model space. For our linear model, the fully Bayesian approach would introduce a prior over $\theta = (\phi, A, B)$ and then perform inference. Unfortunately, this is rarely done. Performing inference at the higher levels of a Bayesian hierarchical model is often too computationally expensive to be practical outside of toy settings. As an alternative, when the number of hyperparameters is small relative to the number over observations, the posterior distribution over hyperparameters may be well approximated by a point mass at its mode $p(\theta|Y) \approx \delta(\theta - \theta_\star)$ with

$$\theta_\star = \arg\max_\theta p(Y; \theta). \tag{2.42}$$

In this setting, the likelihood over hyperparameters dominates the prior, and thus the latter is ignored. Here, the model evidence can provide us with a learning objective to select our hyperparameters. The possibility of performing gradient-based optimisation of $\log p(Y; \theta)$ makes this approach an attractive alternative to traditional cross validation. We must be cautious when using this technique when the dataset is small or our hyperparameter space is large however, as the point-mass-posterior assumption can break, leaving us susceptible to overfitting.

2.4.3 The evidence of the linear model

For the Gaussian linear model, the model evidence can be computed in closed form

$$\log p(Y;\,\phi, B, A) = \log \int_w \sqrt{\frac{\det B}{(2\pi)^n}} \exp\left(-\frac{1}{2}\|Y-\Phi w\|_B^2\right) d\mathcal{N}(0, A^{-1})$$
$$= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det(\Phi A^{-1}\Phi^T + B^{-1})^{-1} - \frac{1}{2}\|Y\|^2_{(\Phi A^{-1}\Phi^T + B^{-1})^{-1}} \tag{2.43}$$

which amounts to the log density of the targets under a multivariate Gaussian with mean 0 and covariance $\Phi A^{-1}\Phi^T + B^{-1}$. The equivalent kernelised form, which can be used to optimise kernel hyperparameters, like the lengthscale, is obtained by again substituting $K = \Phi A^{-1}\Phi^T$. The cost of evaluating (2.43) is cubic in nc because of both the linear solve against $K + B^{-1}$, and because of the appearance of the same matrix's log-determinant. The former appeared in the expression for the posterior distribution (e.g.(2.24)) but the latter presents a new challenge, which we will also tackle in the later chapters of this thesis.

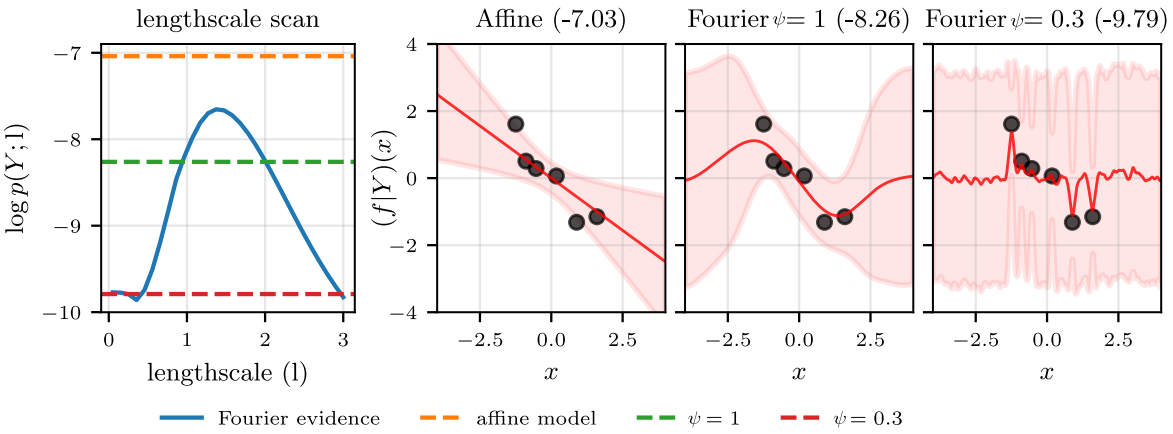


Fig. 2.9 The leftmost plot displays the evidence of a $d = 500$ random Fourier basis function linear model as a function of the lengthscale ψ parameter for the toy 1d dataset of Figure 2.3. The evidence of the Affine model, and the $\psi = 1$ and $\psi = 0.3$ Fourier models are indicated as dashed horizontal lines. The posterior mean function, along with 2 standard deviation errorbars are displayed for each of these models in the three plots on the right. In these, each model's evidence is provided in parenthesis in the plot title. Other hyperparameters match those of Figure 2.3.

Using the Woodbury matrix identity and the matrix determinant lemma we recover the primal form of (2.43), with cost cubic in the number of parameters d

$$-\frac{n}{2}\log(2\pi) + \frac{1}{2}\log\det B - \frac{1}{2}\underbrace{\|Y - \Phi w_\star\|_B^2}_{\text{data fit}} - \frac{1}{2}\underbrace{\|w_\star\|_A^2}_{\text{parameter norm}} - \frac{1}{2}\underbrace{\log\det\frac{H+A}{A}}_{\text{posterior contraction}}. \quad (2.44)$$

This expression more intuitively captures the quality of fit vs simplicity trade-off discussed at the beginning of the section. There is a data fit term that rewards the posterior mean for passing near the targets. There is a prior fit term that ensures the norm of the posterior mean weights are small in the metric given by the prior precision. Finally, the determinant ratio term measures the contraction of the posterior covariance's volume relative to the prior covariance. Up to a constant factor, this quantity matches the information that was gained by our model by seeing the data, in nats. This captures the intuition that the marginal likelihood rewards models that are able to explain the targets well a priori, and thus do not learn much from conditioning on the targets.

Figure 2.9 compares the model evidence for the Affine and random Fourier models introduced in Section 2.1.1 and the dataset from Figure 2.3. The targets are roughly arranged in a straight line, making the affine model a good fit. Although different lengthscale Fourier models can also fit this data, their additional flexibility penalises them; there is no lengthscale setting for which the Fourier model's evidence surpasses the Affine model's. The leftmost plot shows the evidence as a function of the lengthscale. Too small lengthscale values lead to too flexible models that overfit. This is the case for the model in the rightmost plot. Too large lengthscale values would under fit. The optima is somewhere in the middle.

Remark **All linear models are wrong, but the evidence can tell us which are useful**
"All models are wrong, but some are useful" – George Box
We almost never expect the data we are modelling to have been generated via a noisy linear combination of basis functions. On the other hand, we usually judge models on whether their predictions about quantities we care about match empirical outcomes to a desirable tolerance. It is well known that the Bayesian posterior does not provide optimal predictions under model misspecification (Draper and Krnjajic, 2010; Masegosa, 2020). It may seem surprising then, that the linear model's evidence can be shown, using the PAC-Bayes framework, to provide guarantees about generalisation performance (Germain et al., 2016), informing us about whether our models are "useful".

It is worth noting that PAC-Bayes guarantees no longer hold if we use the model evidence for hyperparameter selection. If we select among a discrete set off hyperparameters, we could obtain relaxed guarantees via a union bound, but this would not work for a continuous hyperparameter space.

2.4.4 Effective dimension

We conclude with a discussion of the effective dimension, a quantity intimately related to the evidence of the Gaussian linear model (Mackay, 1992a; Maddox et al., 2020; Wipf and Nagarajan, 2007). We will make heavy use this quantity to derive efficient algorithms for hyperparameter learning in Chapter 6. Let $\lambda_1, \lambda_2, \ldots, \lambda_d$ denote the sorted (in descending order) eigenvalues of the weight space loss curvature $M = \Phi^T B \Phi$. For a regulariser of the form $A = aI$, the effective dimension γ is given by

$$\gamma = \sum_{i=1}^{d} \frac{\lambda_i}{\lambda_i + a}. \tag{2.45}$$

When $\lambda_i \gg a$ the term inside of the sum will roughly be of value 1. When $\lambda_i \ll a$ it will be roughly 0. Thus, the effective dimension $\gamma \in [0, \min(nc, d)]$ counts the number of directions in parameter space which are determined by the data.

Following Mackay (1992a), we can construct a more general definition for the effective dimension that doesn't require an isotropic regulariser, by taking it to be the trace of the matrix that maps the maximum likelihood parameter vector $H^{-1}Y$ into the maximum a posteriori weights $(\Phi^T B \Phi + A)^{-1} Y$. That is,

$$\gamma \coloneqq \text{Tr}\left(\Phi^T B \Phi H^{-1}\right) = d - \text{Tr}\left(A H^{-1}\right) = \text{Tr}\left(K(K + B^{-1})^{-1}\right), \tag{2.46}$$

where we have provided an additional two forms of the quantity, each providing for a complementary interpretation. Using the cyclical property of the trace we can see that the leftmost form is equivalent to $\text{Tr}\left(B\Phi H^{-1}\Phi^T\right)$. That is, the sum of the ratios of the marginal posterior predictive variance to noise variance at the observed inputs. Since each observation reduces the marginal uncertainty in the posterior over functions at that point to at least the corresponding diagonal entry of B, each diagonal entry of $B\Phi H^{-1}\Phi^T$ must be smaller or equal to 1. The degree to which the predictive variance is smaller than the observation noise depends on how well the datapoints explain each other. If they explain each other a lot, i.e. many inputs map to nearby points in the RKHS, the effective dimension decreases.

The middle form of the effective dimension in (2.46) provides us with the same intuition, but through the ratio of the prior and posterior covariance over the weights. The rightmost form is the trace of the matrix that maps the representer weights obtained by fitting the data without regularisation $K^{-1}Y$ onto the representer weights corresponding to the posterior mean function $(K + B^{-1})^{-1}Y$.

Derivation Relating the forms of the effective dimension
We first relate the first and second equalities in (2.46).

$$\begin{aligned}\text{Tr}\left(H^{-1}\Phi^T B\Phi\right) &= \text{Tr}\left((I + A^{-1}\Phi^T B\Phi)^{-1}A^{-1}\Phi^T B\Phi\right)\\ &= \text{Tr}\left(I - (I + A^{-1}\Phi^T B\Phi)^{-1}\right) = d - \text{Tr}\left(AH^{-1}\right).\end{aligned}$$

We now connect the first and third equalities

$$\begin{aligned}\text{Tr}\left(H^{-1}\Phi^T B\Phi\right) &= \text{Tr}\left(B(K - K(K + B^{-1})^{-1}K)\right)\\ &= \text{Tr}\left(BK(I - (KB + I)^{-1}KB)\right) = \text{Tr}\left(BK(KB + I)^{-1}\right)\\ &= \text{Tr}\left(K(K + B^{-1})^{-1}\right).\end{aligned}$$

2.5 Limitations of conjugate Gaussian-linear Bayesian reasoning

We have seen how the linear model with a Gaussian prior over its weights, or Gaussian process, acts as a conjugate prior for the likelihood induced by Gaussian observation noise, providing us with a closed form expression for the Bayesian posterior (2.9), and model evidence (2.43), both Gaussian forms. Alas, conjugacy is quickly lost when constructing more sophisticated Bayesian models that more accurately describe real-world systems of interest. It is lost if we define a non-Gaussian prior over the weights, for instance heavy tailed priors used to model outlier events (West, 2018) or priors designed to favour sparse posteriors, like the horseshoe (Carvalho et al., 2009). It is lost if we use non-Gaussian likelihoods, like the categorical used in classification (Bishop and Tipping, 2003), or the Poisson used to count neural spikes (Heeger, 2000) and X-ray quanta (Elbakri and Fessler, 2003) in computed tomography. Conjugacy is also lost if our model presents a non-linear relationship between its parameters and outputs, for instance due to the use of a linking function that constrains the output range.

Of special interest for this thesis is the use of the neural network function class. These models can be thought of as basis function linear models in which the basis function parameters are treated as model parameters, instead of hyperparameters, and thus inferred from the data. Neural networks are used to model processes where we have little intuition of what the data-generating process might look like, and thus we can not manually choose a set of basis functions. To make up for this lack of prior knowledge, very large and flexible models are paired with vast datasets.

This leads to our second major setback. The closed form expressions of linear model inference involve cubic operations: linear system solves and log-determinant computations, both of which present cubic time complexity. We may choose to either pay this cost in terms of the number of observations times output dimensions $\mathcal{O}\left((nc)^3\right)$ or model parameters $\mathcal{O}\left(d^3\right)$ (when the feature space is finite-dimensional). This provides little consolation in the modern setting where it is common to work with large datasets. For instance, the Imagenet dataset (Russakovsky et al., 2015), which is a benchmark three orders of magnitude smaller than the datasets used to train the largest models in deployment (Dosovitskiy et al., 2021), has $nc \approx 10^9$. The ResNet-50 neural network (He et al., 2016a), another common benchmark model that is around 10 times smaller than the state of the art models, presents a parameter space with $d \approx 25 \cdot 10^6$. One may think that linear models could scaled up to problems of modern interest via efficient numerical linear algebra routines implemented on GPU accelerators. However, at these scales, even storing covariance matrices, whose number of entries are quadratic, becomes intractable due to the $\mathcal{O}\left((nc)^2\right)$ or $\mathcal{O}\left(d^2\right)$ memory cost. For instance storing a covariance matrix for a parameter space the size of ResNet-50's would require around 2500 Terabytes.

The following chapter reviews approximations to Bayesian inference which may be tractably computed when faced with non-conjugacy or large covariance matrices. Unfortunately, we will see how these approximations tend to break down when faced with the neural network model class and real-world sized datasets. The rest of the thesis aims to fill this gap by introducing methods for very large scale Bayesian reasoning with linear models and neural networks.

Chapter 3

Approximate inference methods for linear models and neural networks

The need for approximate inference arises in the linear-Gaussian model when the problem setting becomes too large, making closed form expressions too computationally expensive to evaluate. It also arises when working with non-conjugate Bayesian models. This thesis deals with both settings, 1) Bayesian inference in Gaussian linear models with millions of parameters and observations, and 2) Bayesian inference in neural networks. On our way to tackling these problems, this chapter reviews approximate inference methods for linear models and Gaussian processes, and how these can be extended to neural networks. Section 3.1 covers Variational Inference (VI) in both its parameter-space and inducing point flavours. Section 3.2 covers the use of Conjugate Gradient (CG) methods. Finally, Section 3.3 introduces the Laplace approximation as well as its linearised variant for neural networks. Through different paths, all of these methods provide both an approximation to the posterior as well as the model evidence. We do not delve into Markov Chain Monte Carlo (MCMC) techniques, but instead refer refer to Andrieu et al. (2003) for a general overview and to Neal (1992) for a discussion of their application to neural networks.

3.1 Approximating the posterior distribution: Variational inference

We commence from Bayes rule (2.8). Making the set of hyperparameters $\theta \in \Theta$ explicit in the notation, we take logs on both sides of the equality, and re-arrange it as

$$\log p(Y;\, \theta) = \log p(Y|w;\, \theta) + \log \pi(w;\, \theta) - \log \pi(w|Y;\, \theta), \tag{3.1}$$

to evaluate the evidence. This expression holds for any value of w, allowing us to take expectations on both sides of the equality with respect to any distribution over w. We thus introduce the variational distribution Q, with density $q(w)$ such that $dQ = q(w)d\nu$, and use it to derive the lower bound

$$\begin{aligned} \log p(Y;\, \theta) &= \mathbb{E}_{w\sim Q}\left[\log p(Y|w;\, \theta) + \log \pi(w;\, \theta) - \log \pi(w|Y;\, \theta)\right] && (3.2)\\ &\geq \mathbb{E}_{w\sim Q}\left[\log p(Y|w;\, \theta) + \log \pi(w;\, \theta)\right] + \mathbb{H}\left(Q\right) \coloneqq \mathcal{M}(Q, \theta). && (3.3) \end{aligned}$$

We refer to $\mathcal{M}(Q, \theta)$ as the Evidence Lower BOund (ELBO) and $\mathbb{H}$ is the differential entropy. The inequality is true because the cross entropy can be decomposed into a sum of an entropy and the KL divergence between the distributions being compared, and the latter term is greater or equal to 0. That is, adopting the density-based notation for the KL divergence $\mathrm{KL}(q(w) \parallel \pi(w|Y)) = \int \log \frac{Q}{\Pi_{w|Y}}\, dQ$, we have $\mathbb{E}_{w\sim Q}\left[-\log \pi(w|Y;\, \theta)\right] = \mathbb{H}\left(Q\right) + \mathrm{KL}(q(w) \parallel \pi(w|Y)) \geq \mathbb{H}\left(Q\right)$. Thus, when $\mathrm{KL}(q(w) \parallel \pi(w|Y)) = 0$ and thus the variational posterior matches the Bayesian posterior $q(w) = \pi(w|Y)$, (3.3) becomes an equality, and the ELBO matches the evidence $\log p(Y;\, \theta) = \mathcal{M}(Q, \theta)$.

The ELBO allows us to transform the problem of Bayesian inference into one of variational optimisation. By maximising $\mathcal{M}$ with respect to our variational distribution $Q \in \mathcal{Q}$, we approximate the Bayesian posterior distribution in the sense of minimising $\mathrm{KL}(q(w) \parallel \pi(w|Y))$ (Hinton and van Camp, 1993). We may do this even if our search space, the variational family $\mathcal{Q}$, does not contain the true posterior $\Pi_{w|Y} \notin \mathcal{Q}$. This allows us to tractably approximate the Bayesian posterior even when this distribution is analytically or computationally intractable (Attias, 1999). Evaluating the ELBO does not require conjugacy, only being able to evaluate the log-likelihood function and the prior log-density. We demonstrate this for a 1d toy classification example, where the likelihood is Bernoulli, in Figure 3.1. The expectation in (3.3) is often unbiasedly estimated via Monte Carlo. Thus, the requirements on the variational distribution are that we can sample from it and that we can

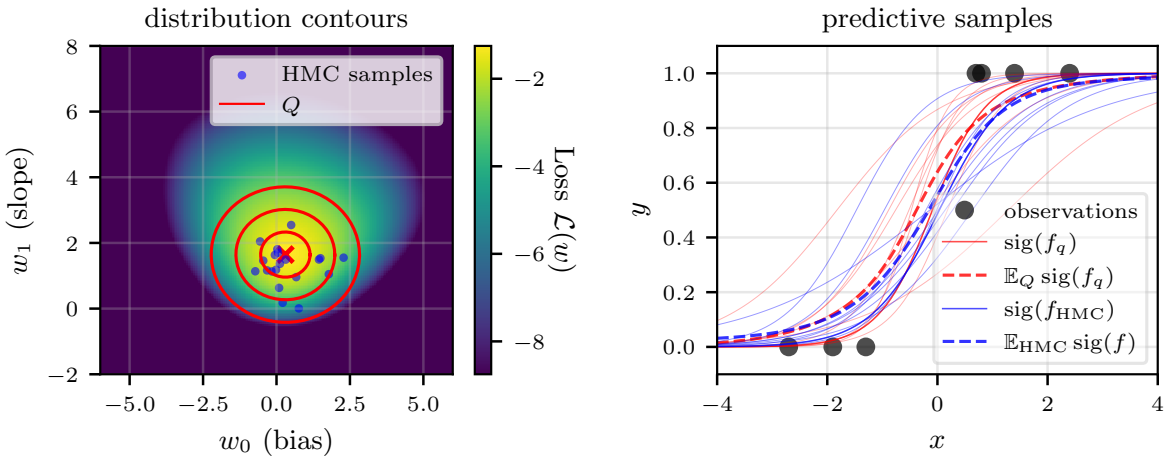


Fig. 3.1 Classification example with our affine linear model, where we place a Gaussian prior over the weights, we use a sigmoid linking function and a Bernoulli likelihood. The left plot shows how the loss landscape, which up to a constant matches the log posterior density, presents a non quadratic form; the top of the distribution is wider than the bottom. We approximate this posterior with a Gaussian variational distribution Q and with Hamiltonian Monte Carlo (HMC). In this setting, only the latter method provides an unbiased approximation. Despite this, the plot on the right shows how both approximations lead to similar predictions. However, the variational approximation places more mass on low slope functions, resulting in slight underestimation of the steepness of the sigmoid.

compute its entropy. Relaxing the latter constraint is an active area of research (Titsias and Ruiz, 2019; Uppal et al., 2023).

Remark Protection against overfitting
It is often said that variational parameters are protected against overfitting. This is because optimising the ELBO with respect to these parameters always brings the variational distribution closer to the true posterior. Thus, choosing a more flexible variational family that leads to a tighter ELBO should always lead to a better posterior approximation. Unfortunately, the same is not true about the model hyperparameters, whose optimisation with the ELBO can lead to overfitting (see, for instance, Ober et al. (2021)).

The ELBO can also act as a hyperparameter selection objective, acting as a substitute for the model evidence when the later is not tractable. However, if the variational posterior differs from the true posterior, the hyperparameter learning objective will be biased (see, for instance, Turner and Sahani (2011)). We illustrate this bias in Figure 3.2. The variational EM algorithm (Bishop, 2006; Dempster et al., 1977; Neal and Hinton, 1998) implements this

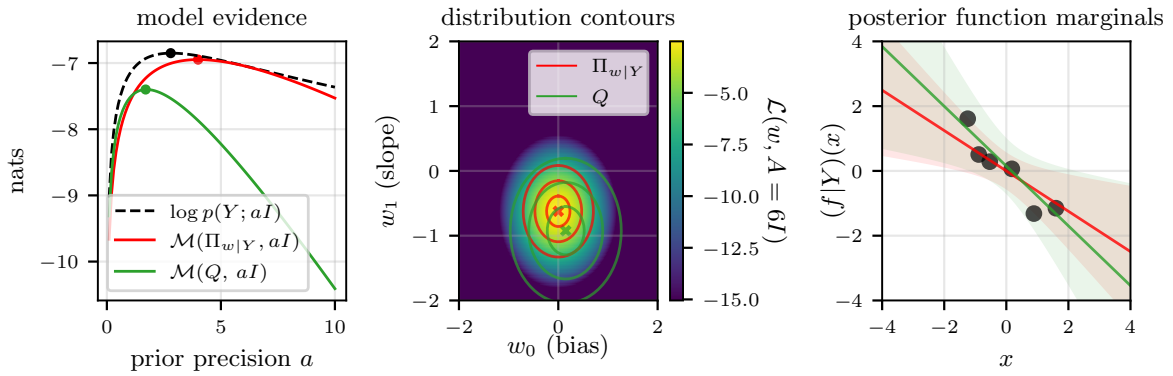


Fig. 3.2 Variational inference in the Gaussian affine linear model, fit to the toy dataset dataset in Figure 2.3. The leftmost plot shows the model evidence as a function of the isotropic prior covariance $A = aI$. We also display an ELBO where the variational posterior is set to the true posterior when $a = 6$, denoted $\Pi_{w|Y}$ in the plot. The bound is tight at $a = 6$, as predicted by (3.3). However, since the posterior over the weights does not change as we scan a, the optima of the ELBO, marked with a red dot, differs from the optima of the evidence. Hyperparameter selection with this objective would be biased. We also display, in green, the ELBO corresponding to a different variational posterior Q. Since Q, doesn't match the true posterior for any value of a, the bound is never tight. It is also a biased estimate of the evidence. The middle plot shows the loss function when a is set to 6 as well as the 1, 2 and 3 standard deviation contours for the log-density of $\Pi_{w|Y}$ and Q. Finally, the rightmost plot shows the mean and 2 standard deviation errorbars of the posterior distribution over functions corresponding to each of the 2 variational posteriors under consideration.

idea by iterating variational posterior optimisation and hyperparameter optimisation steps: 1) setting $Q = \arg\max_{Q \in \mathcal{Q}} \mathcal{M}(Q, \theta)$ in the E step and 2) $\theta = \arg\max_{\theta \in \Theta} \mathcal{M}(Q, \theta)$ in the M step. If $\Pi_{w|Y} \in \mathcal{Q}$, the E step will attain the exact posterior and the EM algorithm is guaranteed to not decrease the model evidence. Alternatively, one may optimise $\mathcal{M}$ with respect to $\{Q, \theta\}$ jointly using gradient-based optimisation.

Remark The dangers of model comparison with the ELBO
The ELBO is not a reliable tool for model comparison. If a model obtains a larger ELBO than another, it is not guaranteed to have a larger evidence. The model with the smaller ELBO could have a larger evidence and the difference in ELBO values could be due to there being more slack in the second model's bound.

Beyond approximate inference in predictive models, the ELBO also plays an important role in information theory and data compression; we refer to Flamich (2019); Hinton and van Camp (1993) and chapter 33 of MacKay (2003) for in-depth discussion.

3.1.1 VI in the parameter space of the linear model

We now provide the explicit form of the ELBO for the weight-space Gaussian linear model introduced in (2.2) paired with a multivariate Gaussian variational family $Q = \mathcal{N}(w_q, \Sigma_q)$ with variational parameters $w_q \in \mathbb{R}^d$ and $\Sigma_q \in \mathbb{R}^{d\times d}$. In this case, the true posterior is contained within the variational family. The ELBO is

$$\mathcal{M}(w_q, \Sigma_q, A, B, \phi) = \frac{1}{2}\mathbb{E}_{w\sim\mathcal{N}(w_q,\Sigma_q)}\big[- n\log(2\pi) - \log\det B^{-1} - \|Y - \Phi w\|_B^2 \\ - \log\det A^{-1} - \|w\|_A^2 + \log\det\Sigma_q + d\big], \quad (3.4)$$

where we have substituted Q for its variational parameters, which uniquely define the distribution, in the ELBO's arguments. Evaluating the expectation we obtain

$$\mathcal{M}(w_q, \Sigma_q, A, B, \phi) = \frac{1}{2}\big(- n\log(2\pi) - \log\det B^{-1} - \log\det A^{-1} - \|w_q\|_A^2 - \mathrm{Tr}(\Sigma_q A) \\ - \|Y - \Phi w_q\|_B^2 - \mathrm{Tr}(\Phi\Sigma_q\Phi^T B) + \log\det\Sigma_q + d\big). \quad (3.5)$$

This expression will be of particular interest in Chapter 5 and Chapter 6, where we will use the Laplace approximation to the posterior, a multivariate Gaussian, as the variational distribution for large scale models. The variational posterior distribution over functions is computed analogously to (2.9) by substituting the Bayesian weight posterior with its approximation

$$f_q(\cdot) = \phi(\cdot)w \quad \text{with} \quad w \sim \mathcal{N}(w_q, \Sigma_q). \quad (3.6)$$

3.1.2 VI in function space: inducing points

We now look at the dual form of variational inference for linear models where the approximate distribution is specified directly over function outputs. To this end, we introduce an array of m inducing points $Z = (z_1, z_2, \ldots, z_m)$ with $z_i \in X$. The variational inducing point framework of Titsias (2009a,b) substitutes our observed targets Y with the inducing targets $U \in \mathbb{R}^{cm}$, each of which is associated with an inducing point. We start by constructing a Gaussian process conditioned on the set of inducing locations and targets

$$(f^{(Z)}|U) \sim \mathrm{GP}(\mu^{(Z)}_{f|U}, k^{(Z)}_{f|U}), \quad (3.7)$$

where the superscript notation $^{(Z)}$ makes explicit that the input locations correspond to Z and not X. The mean and covariance functions are given by

$$\mu^{(Z)}_{f|U}(\cdot) = K_{(\cdot)Z}K^{-1}_{ZZ}U \quad k^{(Z)}_{f|U}(\cdot,\cdot') = K_{(\cdot,\cdot')} - K_{(\cdot)Z}K^{-1}_{ZZ}K_{Z(\cdot')}, \tag{3.8}$$

where $[K_{ZZ}]_{ij} = k(z_i, z_j)$, $i, j \leq m$ and we again use $K_{Z(\cdot)}$ for the stacked evaluation functionals $k(z_i, \cdot)$, $i \leq m$. These expressions match (2.24), with the observed inputs X and targets Y replaced by the inducing inputs Z and inducing targets U.

We now place a multivariate Gaussian variational distribution over the inducing targets $Q = \mathcal{N}(u^{(Z)}_q, K^{(Z)}_q)$, with $u^{(Z)}_q \in \mathbb{R}^{cm}$ and $K^{(Z)}_q \in \mathbb{R}^{cm\times cm}$. Following Titsias (2009a), we choose the mean and covariance of this distribution that minimises the KL divergence between the variational Gaussian process $\mathbb{E}_{U\sim Q}[f^{(Z)}|U]$ and the posterior Gaussian process $f|Y$ (Matthews et al., 2016)[1]. These are

$$u^{(Z)}_q = K_{ZZ}(K_{ZZ} + K_{ZX}BK_{XZ})^{-1}K_{ZX}BY \tag{3.9}$$

$$K^{(Z)}_q = K_{ZZ}(K_{ZZ} + K_{ZX}BK_{XZ})^{-1}K_{ZZ}, \tag{3.10}$$

where $[K_{XZ}]_{ij} = k(x_i, z_j)$, $i < n\, j < m$. Using this, we marginalise out the inducing targets in (3.7), arriving at the optimal variational Gaussian process

$$(f^{(Z)}|Y) \sim \text{GP}(\mu^{(Z)}_{f|Y}, k^{(Z)}_{f|Y}), \tag{3.11}$$

with mean and covariance functions

$$\mu^{(Z)}_{f|Y}(\cdot) = K_{(\cdot)Z}(K_{ZZ} + K_{ZX}BK_{XZ})^{-1}K_{ZX}BY \tag{3.12}$$

$$k^{(Z)}_{f|Y}(\cdot,\cdot') = K_{(\cdot,\cdot')} + K_{(\cdot)Z}((K_{ZZ} + K_{ZX}BK_{XZ})^{-1} - K^{-1}_{ZZ})K_{Z(\cdot')}. \tag{3.13}$$

These expressions contain linear solves against K_{ZZ} instead of K. The number of inducing points is typically chosen to be smaller than the number of observations $m < n$ and thus the cost is lowered from $\mathcal{O}\left((nc)^3\right)$ to $\mathcal{O}\left((mc)^3\right)$.

[1] In practise, this KL divergence between stochastic processes can be minimised by minimising the KL divergences between the multivariate Gaussians given by evaluating the variational GP and posterior GP at the set of observed and inducing inputs $\{X, Z\}$ jointly.

Connecting inducing points to the Nyström approximation

The expressions (3.12) and (3.13) match those that we obtain if we substitute our Gaussian process prior with $\mathrm{GP}(0, K_{(\cdot),Z}K_{ZZ}^{-1}K_{Z,(\cdot')})$, and proceed with exact GP inference, as in Section 2.2.3. With this, every instance of K is replaced with $K_{XZ}K_{ZZ}^{-1}K_{ZX}$, revealing that the variational Gaussian process amounts to a Nyström approximation of the kernel matrix (Wild et al., 2021).

Derivation Nyström pathwise representation of the optimal variational GP
To show the connection between the Nyström approximation and variational inducing point GPs, we leverage the pathwise formulation of the GP random function (2.28) but replace every instance of with $K_{XZ}K_{ZZ}^{-1}K_{ZX}$, yielding

$$\begin{gathered}(f^{(Z)}|Y)(\cdot) = f(\cdot) + K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}(K_{XZ}K_{ZZ}^{-1}K_{ZX} + B^{-1})^{-1}(Y - f^{(Z)}(X) - \varepsilon) \\ \varepsilon \sim \mathcal{N}(0, B^{-1}) \qquad f \sim \mathrm{GP}(0, k) \qquad f^{(Z)}(\cdot) = K_{(\cdot)Z}K_{ZZ}^{-1}f(Z).\end{gathered} \tag{3.14}$$

We now check the correctness of this expression by calculating the moments of this Gaussian process' marginal distributions and show them to match those of the KL-optimal variational Gaussian process given in (3.12) and (3.13). Write

$$\begin{aligned}\mathbb{E}[(f^{(Z)}|Y)(\cdot)] &= K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}(K_{XZ}K_{ZZ}^{-1}K_{ZX} + B^{-1})^{-1}Y && (3.15)\\ &= K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}B(K_{XZ}K_{ZZ}^{-1}K_{ZX}B + I)^{-1}Y && (3.16)\\ &= K_{(\cdot)Z}(K_{XZ}BK_{XZ} + K_{ZZ})^{-1}K_{ZX}BY && (3.17)\\ &= \mu_{f|Y}^{(Z)}(\cdot) && (3.18)\end{aligned}$$

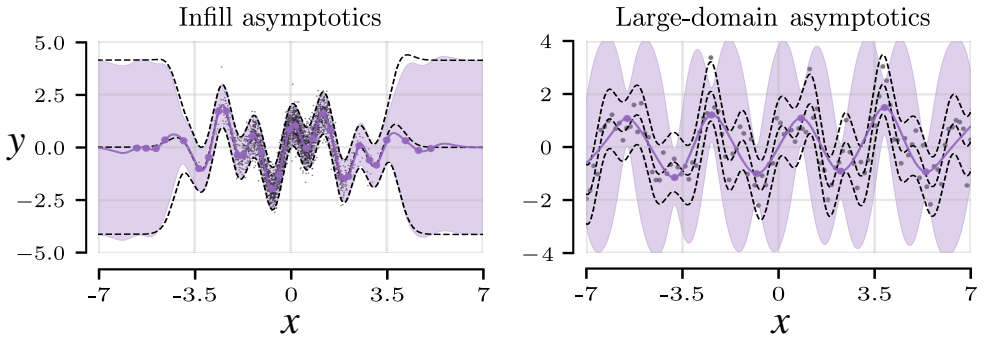


Fig. 3.3 Illustration of variational inducing point GP inference with a squared exponential kernel on 10k datapoints from $\sin(2x)+\cos(5x)$ with observation noise distribution $\mathcal{N}(0, 0.5)$. The inducing point locations are marked with purple dots. All variational parameters are fit with SVGP (3.27). The true GP posterior is marked with a think black dashed line. Contours denote 2 standard deviation credible intervals for the predictive posterior. *Infill asymptotics* considers $x \sim \mathcal{N}(0, 1)$. A large number of points near zero result in a very ill-conditioned kernel matrix. VI can summarise the data with only 20 inducing points. *Large domain asymptotics* considers data on a regular grid with fixed spacing. Note that most of the data is not visible in the plot. This problem is better conditioned. However, 1024 inducing points are not enough to summarise the data, leading to poor performance.

and

$$\begin{aligned}
&\mathrm{Cov}((f^{(Z)}|Y)(\cdot) - \mu^{(Z)}_{f|Y}(\cdot)) && (3.19)\\
&= \mathbb{E}((f^{(Z)}|Y)(\cdot) - \mu^{(Z)}_{f|Y}(\cdot), (f^{(Z)}|Y)(\cdot') - \mu^{(Z)}_{f|Y}(\cdot')) && (3.20)\\
&= K_{(\cdot,\cdot')} - K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}(K_{XZ}K_{ZZ}^{-1}K_{ZX} + B^{-1})^{-1}K_{XZ}K_{ZZ}^{-1}K_{Z(\cdot')} && (3.21)\\
&= K_{(\cdot,\cdot')} + K_{(\cdot)Z}K_{ZZ}^{-1}\left(-I + I - K_{ZX}B(K_{XZ}K_{ZZ}^{-1}K_{ZX}B + I)^{-1}K_{XZ}K_{ZZ}^{-1}\right)K_{Z(\cdot')} && (3.22)\\
&= K_{(\cdot,\cdot')} + K_{(\cdot)Z}K_{ZZ}^{-1}\left(-I + (K_{ZX}BK_{XZ}K_{ZZ}^{-1} + I)^{-1}\right)K_{Z(\cdot')} && (3.23)\\
&= K_{(\cdot,\cdot')} + K_{(\cdot)Z}\left(-K_{ZZ}^{-1} + (K_{ZX}BK_{XZ} + K_{ZZ})^{-1}\right)K_{Z(\cdot')} && (3.24)\\
&= k^{(Z)}_{f|Y}(\cdot,\cdot') && (3.25)
\end{aligned}$$

which recovers (3.12) and (3.13), as claimed.

This relationship allows us to gain intuition about the properties of inducing point approximations. These will work well when the conditioning number of K is large. Intuitively, if multiple observed inputs are similar they can be modelled with a single inducing point and analogously if multiple rows of K nearly linearly dependent, their action can be captured by a single row of K_{ZZ}. On the other hand, a dataset where different inputs map

to distant points in the RKHS will be poorly approximated by $m < n$ inducing points. We illustrate these properties in Figure 3.3.

Hyperparameter learning with inducing points

Titsias (2009a) uses the optimal variational GP, given in (3.11), to construct the ELBO

$$\begin{aligned}\mathcal{M}(Z,\theta) = &-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det\left(B^{-1} + K_{XZ}K_{ZZ}^{-1}K_{ZX}\right)\\ &-\frac{1}{2}\|Y\|^2_{(B^{-1}+K_{XZ}K_{ZZ}^{-1}K_{ZX})^{-1}} - \frac{1}{2}\operatorname{Tr}\left(B(K - K_{XZ}K_{ZZ}^{-1}K_{ZX})\right),\end{aligned} \tag{3.26}$$

where the inducing target values are marginalised, leaving the inducing point locations Z as the only variational parameters to be optimised together with the hyperparameters θ.

Two limitations of the bound in (3.26) are that it's data-fit term can not be decomposed into a sum of each observation's contributions, precluding minibatch estimators, and that it is only valid for Gaussian likelihoods. Hensman et al. (2013) addresses both of these issues by introducing the ELBO

$$\begin{aligned}\mathcal{M}(u_q^{(Z)}, K_q^{(Z)}, Z, \theta) =&\mathbb{E}_{\substack{(f^{(Z)}|U)\sim\mathrm{GP}(\mu_{f|U}^{(Z)}, k_{f|U}^{(Z)})\\ U\sim\mathcal{N}(u_q^{(Z)}, K_q^{(Z)})}}\log p\left(Y|(f^{(Z)}|U)\right)\\ &+\mathrm{KL}(\mathcal{N}(u_q^{(Z)}, K_q^{(Z)}) \parallel \mathcal{N}(0, K_{ZZ})),\end{aligned} \tag{3.27}$$

where $p\left(Y|(f^{(Z)}|U)\right)$ is the conditional density of the targets given the variational GP. (3.27) can be shown to be a lower bound on (3.26). Here, the mean and covariance of the variational distribution over U are left as variational parameters to be optimised. However, learning a quadratic number of parameters for the covariance can lead to numerical instability. GPs fit with (3.27) are often referred to as Stochastic Variational Gaussian Processes (SVGP).

3.1.3 Expectation propagation and non-KL divergences

So far, we have discussed algorithms that choose the variational posterior such that its KL divergence to the Bayesian posterior is minimised. However, there is a rich literature that studies the minimisation of other divergences. We only review these works briefly, as they play no role in the later chapters of this thesis.

The power expectation propagation (EP) algorithm (Minka, 2004, 2007) targets the alpha-divergence between a variational posterior, built as a series of site approximations,

one for each observation, and the true posterior. Power EP is a generalisation of regular EP (Minka, 2001; Opper and Winther, 2005) with the latter targeting reverse KL divergences at each site. In turn, EP can be understood as a generalisation of the belief propagation algorithm (Pearl, 1982, 1988). Hernández-Lobato and Adams (2015) extended the EP framework to neural networks, developing an algorithm coined "probabilistic backpropagation". Furthering this line of work, Hernández-Lobato et al. (2016) applied alpha divergences to black box variational inference problems, doing away with the EP framework. Li (2018) extends variational inference to target the family of Rényi divergences (Rényi, 1961), which also generalise the KL-divergence. EP can also be shown to target a dual of the variational lower bound (Li, 2018). This idea has been used to construct hybrid algorithms, which may present better properties for hyperparameter optimisation (Adam et al., 2021; Li et al., 2023).

3.1.4 Variational inference for neural networks and its limitations

Neural networks present very high dimensional and strongly multimodal posterior distributions. This has made it difficult to develop variational inference methods for neural networks that effectively navigate the trade-off between scalability and accuracy of approximation.

The most common choice of variational distribution is a Gaussian that factorises across dimensions[2]. This choice allows for simple implementation and is relatively computationally inexpensive. As a result, it has persisted from the first works on VI for neural networks (Hinton and van Camp, 1993; Saul and Jordan, 1998) to more modern approaches (Blundell et al., 2015; Graves, 2011). However, it can be shown that modelling dependencies between posterior weights is necessary to obtain calibrated uncertainty estimates (Foong et al., 2020).

There have been efforts to leverage more flexible variational distributions. Louizos and Welling (2017) use normalising flows as variational approximations. Dusenberry et al. (2020) target multiple posterior modes with rank-1 Gaussian approximations. Ober and Aitchison (2021) construct an inducing-point based variational distribution with autoregressive structure across layers. On the other hand, Gal and Ghahramani (2016) and Antorán et al. (2020) obtain scalability to very large neural networks by using very crude variational distributions that consist of randomly zeroing subsets of network weights, and network layers, respectively. Another family of approaches re-cast popular optimisation algorithms, like Adam, as variational inference (Khan et al., 2018; Khan and Rue, 2023; Osawa et al., 2019). Unfortunately, despite these efforts, variational methods often reach solutions that underperform traditional maximum likelihood learning of NN parameters in terms of predictive

[2]Factorised approximations are also referred to as *mean field* approximations.

accuracy (Ashukha et al., 2020; Wenzel et al., 2020) or underestimate predictive uncertainty (Foong et al., 2019a, 2020).

3.2 Approximating the posterior computation: Conjugate Gradients

As we saw in Chapter 2, the main impediment to posterior inference in the Gaussian linear model is having to solve large systems of linear equations (see (2.9) (2.24) (2.28)). These present time complexity $\mathcal{O}\left((nc)^3\right)$ and memory complexity $\mathcal{O}\left((nc)^2\right)$ in kernelised form and the same complexity, but in the number of parameters d, when dealing with the weight-space form. The most widely used algorithm to solve linear systems, both in the context of GPs (Artemev et al., 2021; Gibbs and MacKay, 1996; Wang et al., 2019), and also more generally (Boyd and Vandenberghe, 2014; Press et al., 2007), is Conjugate Gradients (CG).

CG is an iterative algorithm. Given the system $(K + B^{-1})^{-1}Y$, CG performs a single matrix-vector product $(K + B^{-1})Y$, with cost $\mathcal{O}\left((nc)^2\right)$, at each iteration. The algorithm recovers the exact solution after at most nc steps, asymptotically recovering the cubic cost. However, the algorithm often converges much faster, delivering very accurate approximations of the linear system solution after only a few iterations. The speed of convergence depends on system conditioning, which we discuss in detail in Section 3.2.2.

3.2.1 Hyperparameter learning with CG

Optimising linear model hyperparameters with the marginal likelihood requires both solving linear systems against the loss Hessian matrix and also computing its log-determinant. Although, we can not compute the log-determinant with CG, we can can compute its gradient as $\partial_\theta \log \det(K + B^{-1}) = \operatorname{Tr}\left((K + B^{-1})^{-1}\partial_\theta(K + B^{-1})\right)$. We now apply Hutchinson (1990)'s trick to substitute the trace with an expectation, obtaining

$$\begin{aligned}\partial_\theta \log p(Y;\, \theta) =& \frac{1}{2}\mathbb{E}_{z\sim\mathcal{N}(0,I_{nc})} z^T (K + B^{-1})^{-1}\partial_\theta(K + B^{-1})z \\ &+ \frac{1}{2}Y^T(K + B^{-1})^{-1}\partial_\theta\left(K + B^{-1}\right)(K + B^{-1})^{-1}Y. \end{aligned} \tag{3.28}$$

The above expression is approximated by constructing a MC estimator of the expectation. Evaluating each MC sample requires a linear solve against $K + B^{-1}$. It is straight forward to apply the same trick to the primal form of the model evidence. This approach, which

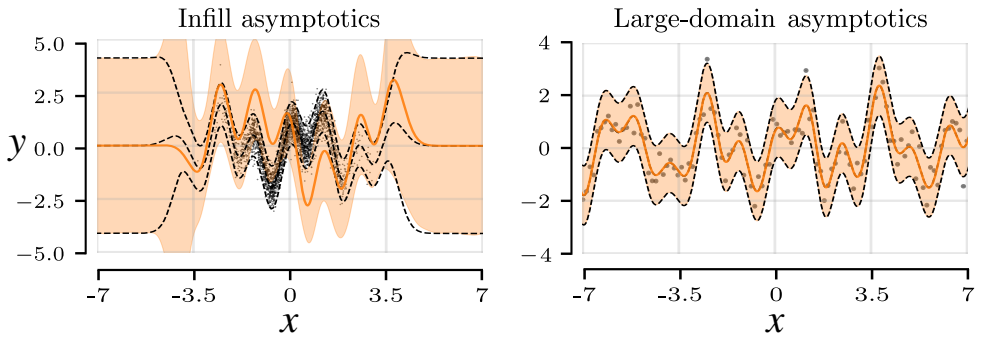


Fig. 3.4 Illustration of variational inducing point GP inference with a squared exponential kernel on 10k datapoints from $\sin(2x)+\cos(5x)$ with observation noise distribution $\mathcal{N}(0, 0.5)$. The inducing point locations are marked with purple dots. All variational parameters are fit with SVGP (3.27). The true GP posterior is marked with a think black dashed line. Contours denote 2 standard deviation credible intervals for the predictive posterior. *Infill asymptotics* considers $x \sim \mathcal{N}(0, 1)$. A large number of points near zero result in a very ill-conditioned kernel matrix, preventing CG from converging (we draw 2000 posterior by CG for 10 minutes on an RTX 2070 GPU.). *Large domain asymptotics* considers data on a regular grid with fixed spacing. Note that most of the data is not visible in the plot. This problem is better conditioned, allowing CG to recover the exact solution.

was first used by Gibbs and MacKay (1996), has become the most popular approximation for hyperparameter learning with large-scale GPs (Gardner et al., 2018). CG for linear model hyperparameter learning can been paired with preconditioning and low precision computation (Maddison et al., 2016) or variational lower bounds (Artemev et al., 2021) to reduce time-to-convergence.

3.2.2 Limitations of Conjugate Gradient inference

The chief limitation of CG is that its convergence speed decreases as the matrix we are solving against becomes more ill-conditioned. Given the system $(K + B^{-1})^{-1}Y$, the number of matrix-vector products needed to guarantee convergence of CG to within a tolerance of ε is

$$\mathcal{O}\left(\sqrt{\text{cond}(K + B^{-1})}\log\frac{\text{cond}(K + B^{-1})\|Y\|}{\varepsilon}\right) \tag{3.29}$$
$$\text{with} \quad \text{cond}(K + B^{-1}) = \frac{\lambda_{\max}(K + B^{-1})}{\lambda_{\min}(K + B^{-1})},$$

where $\lambda_{\max}(K + B^{-1})$ and $\lambda_{\min}(K + B^{-1})$ are the maximum and minimum eigenvalues of $K + B^{-1}$. See (Terenin et al., 2023) for further discussion on (3.29). Although CG performs well in many GP use cases, for instance (Gardner et al., 2018; Wang et al., 2019),

the condition number cond$(K+B^{-1})$ need not be bounded, and conjugate gradients may fail to converge quickly (Terenin et al., 2023). We illustrate this in Figure 3.4. Nonetheless, by exploiting the quadratic structure of the objective, substantially better worst-case convergence rates can be shown for CG than alternatives, like gradient descent (Blanchard and Krämer, 2010; Zou et al., 2021). This makes the results of Chapter 4, where we show that SGD can be used to approximate GP posteriors notably faster than alternative methods, surprising.

3.3 Approximating the function class: the linearised Laplace approximation

The Laplace approximation is a classical technique in Bayesian statistics for constructing Gaussian surrogates for analytically intractable posterior distributions. The Laplace approximation was first applied to neural networks by MacKay (1992a). We will also focus on the neural network setting here, as it is of primary interest for the rest of this thesis, forming the basis of Chapter 5, Chapter 6 and Chapter 7. In doing this, we will also see how the Laplace approximation can be applied to non-conjugate linear models which can be seen as a particular case of neural networks.

Let the the function $g : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}^c$ be a neural network and $v \in \mathcal{V} \subseteq \mathbb{R}^d$ refer to its parameters, flattened into a single vector. We train it to solve a c output prediction problem, by minimising a loss of the form

$$\mathcal{L}_g(v) = \sum_{i=1}^{n} \ell(y_i, g(v, x_i)) + \mathcal{R}(v), \tag{3.30}$$

where the subscript in $\mathcal{L}_g$ makes explicit that our model is the NN g, ℓ is a data fit term (a negative log-likelihood) which we assume to include any linking functions, and $\mathcal{R}$ is a regulariser. We do not assume either to be quadratic. This procedure returns the weights $v_\star \in \arg\min_{v\in\mathbb{R}^d} \mathcal{L}_g(v)$ [3].

Notation for gradients and Hessians We use m $\partial_v^m[g(v,x)](v')$ to denote the mth order mixed partial derivatives of g with respect to v evaluated at (v', x). We use $\partial_x^m f(x')$ to refer to $\partial_x^m[f(x)](x')$ for single argument functions, where no ambiguity exists.

[3] We use $\in$ since NN loss functions are almost always multimodal and thus there exist a set of multiple minimisers.

With that, the Laplace method constructs a locally quadratic approximation to $\mathcal{L}_g$ around the mode

$$\mathcal{L}_g(v) = \mathcal{L}_g(v_\star) + \frac{1}{2}\|v - v_\star\|^2_{\partial_v^2 \mathcal{L}_g(v_\star)} + \mathcal{O}(v^3), \tag{3.31}$$

where the first order term cancels since $\partial_v \mathcal{L}_g(v_\star) = 0$ and $\partial_v^2 \mathcal{L}_g(v_\star) \in \mathbb{R}^{d \times d}$ is the Hessian of the loss at $v_\star$. We use this quadratic approximation to the loss to define the negative log density of an approximate posterior, which by inspection corresponds to the Gaussian

$$\mathcal{N}\left(v_\star,\, (\partial_v^2 \mathcal{L}_g(v_\star))^{-1}\right). \tag{3.32}$$

Remark The asymptotic exactness of the Laplace approximation
The Bernstein–von Mises theorem tells us that for any likelihood function ℓ and under relatively weak conditions, the posterior distribution converges to a Gaussian centred at the maximum likelihood parameter setting as the number of observations goes to infinity, i.e. $n \to \infty$ (Bernstein, 1946; Walker, 1969). This result yields credence to the Laplace approximation in big-data settings. Indeed, MacKay (1992a) reports increased approximation accuracy for larger number of observations.

Despite the Laplace approximation being the first method developed for Bayesian reasoning with NNs (MacKay, 1992a), modern adaptions of the method, some of which are introduced in Chapter 5 and Chapter 6, represent the state-of-the-art in the field of Bayesian deep learning (Antorán et al., 2023; Daxberger et al., 2021a). The method has also seen success when applied to non-conjugate linear models, where the likelihood is non-Gaussian (Rue et al., 2009). We go on to discuss the use of the Laplace approximation, in its linearised variant, for predictive variance estimation and for model evidence approximation in neural networks.

3.3.1 Linearising our network at prediction time

Despite the closed form of the Laplace posterior over NN parameters, integrating out the parameters to evaluate the posterior distribution over functions $g(v, \cdot)$, $v \sim \mathcal{N}(v_\star,\, (\partial_v^2 \mathcal{L}_g(v_\star))^{-1})$ remains analytically intractable. MacKay (1992a) resolves this by introducing an additional approximation: a local linearisation of the neural network function around $v_\star$. We also do

this, introducing the affine model $h : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}^c$, which performs the map

$$h(w, x) \coloneqq g(v_\star, x) + J(x)(w - v_\star) \tag{3.33}$$

where $J(x_i) \coloneqq \partial_v[f(v, x_i)](v_\star) \in \mathbb{R}^{c \times d}$ is the Jacobian of the NN function evaluated at x with respect to its weights, and we denote the approximate model's parameters as $w \in \mathbb{R}^d$ to highlight their linear relationship with the output. With this, the marginals of the posterior distribution over functions $h(w, \cdot)$, $w \sim \mathcal{N}(v_\star, (\partial_v^2 \mathcal{L}_g(v_\star))^{-1})$ become closed form and Gaussian

$$\mathcal{N}(g(v_\star, x'),\ J(x')(\partial_v^2 \mathcal{L}_g(v_\star))^{-1} J(x')^T). \tag{3.34}$$

Here, we have used that the expectation of an affine transform of a Gaussian random variable is the affine transformation of the mean, and since the mean is $v_\star$, the first order term in (3.33) cancels, leaving only $g(v_\star, x')$.

MacKay (1992a) makes one final approximation. He substitutes the Hessian of the data-fit loss $\partial_v^2[\ell(y, g(v, x))](v_\star)$ for the Generalised Gauss Newton matrix (GGN) $J(x)^T \partial_{\hat{y}}^2 \ell(y, \hat{y}) J(x)^T$ evaluated at the MAP predictions $\hat{y}_i = g(v_\star, x_i)$. With this, the precision of the Laplace posterior becomes

$$\sum_{i=1}^{n} \underbrace{J(x_i)^T \partial_{\hat{y}_i}^2 \ell(y_i, \hat{y}_i) J(x_i)^T}_{\text{GGN}} + \underbrace{\partial_v^2 \mathcal{R}(v_\star)}_{\text{Hessian of regulariser}} \tag{3.35}$$

where $\partial_v^2 \mathcal{R}(v_\star) \in \mathbb{R}^{d \times d}$ is the Hessian of the regulariser, and $\partial_{y_i}^2 \ell(y_i, \hat{y}_i) \in \mathbb{R}^{c \times} c$ is the GGN corresponding to the contribution of each observation to the likelihood.

Remark Comparing the Hessian and the GGN
Using the chain rule of the product, we can decompose the Hessian into the GGN, which captures the curvature of the likelihood function but linearises the NN function, and a second term consisting of the gradient of the log-likelihood multiplied with the Hessian of the NN function

$$\partial_v^2[\ell(y, g(v, x))](v_\star) = \underbrace{J(x)^T \partial_{\hat{y}}^2 \ell(y, \hat{y}) J(x)^T}_{\text{GGN}} + \partial_{\hat{y}}[\ell(y, \hat{y})](g(v_\star, x)) \partial_v^2[g(v, x)](v_\star). \tag{3.36}$$

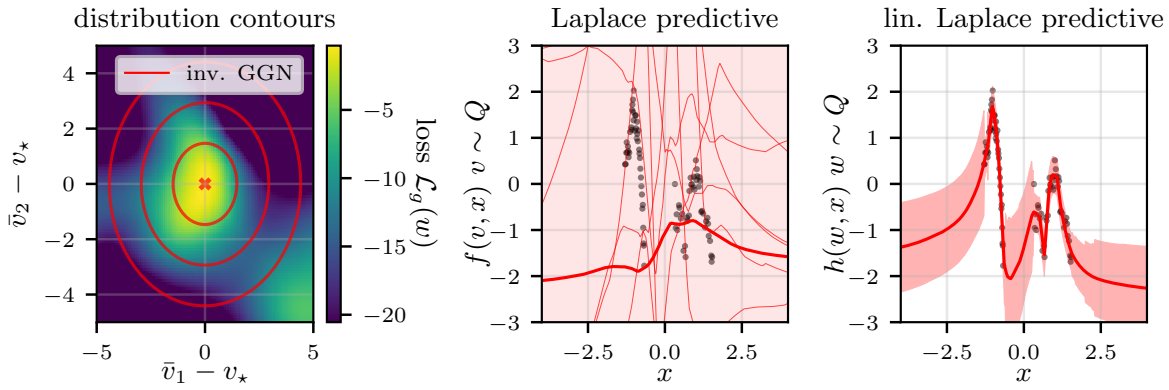


Fig. 3.5 Left: 2d projection of a neural network loss landscape around a mode $v_\star$. We also display the 1, 2 and 3 standard deviation contours of the linearised Laplace (i.e. using the GGN approximation to the Hessian) posterior computed at the mode. Middle: we push Laplace posterior through the NN function and display mean and 2 standard deviation credible regions of the posterior predictive distribution. These do not fit the data. We also display the functions corresponding to 4 posterior samples. Right: the linearised Laplace predictive distribution fits the data well and provides sensible errorbars (2 standard deviation credible regions of the posterior predictive distribution).

From this, we can see that the GGN will be a good approximation to the Hessian when the gradient of the data-fit loss $\partial_{\hat{y}}[\ell(y, \hat{y})](g(v_\star, x))$ is small. It will be exact when we are at an optima of the fit term. For instance, when the NN parametrises the mean of a Gaussian likelihood and the NN output perfectly interpolates the training targets.
Unlike the exact Hessian, the GGN is guaranteed to be PSD. This makes it often preferred in the second order optimisation literature (Becker and LeCun, 1989; Martens, 2014; Schraudolph, 2002), since negative curvature results in linear system solutions lying at infinity, causing optimisers to diverge. Furthermore, the GGN is cheaper to compute than the full Hessian and better lends itself to efficient block-wise approximations. Examples of the latter are the iLQR algorithm (Bemporad et al., 2002) and the Kronecker factored approximation (Martens and Grosse, 2015).

Lawrence (2000) found that the Laplace approximation, without the linearisation step, resulted in very poor quality predictive distributions that did not even assign high density to the train targets. Ritter et al. (2018) make a similar observation, but ameliorate the issue by introducing additional hyperparameters that decrease the variance of the posterior over the weights. We reproduce this result in Figure 3.5. We also show how the true NN posterior can present strongly non-Gaussian features near a mode, leading the Laplace approximation to place some of its mass in very low density regions of the true posterior. It is this that causes

poor predictions. However, local linearisation resolves the issue. This incongruence was resolved recently, roughly 30 years after the publication of Mackay (1992a), by the modern formulation of the linearised Laplace approximation (Antorán et al., 2022; Immer et al., 2021b; Khan et al., 2019b), which we describe in the next section.

3.3.2 A modern view of linearised Laplace

We now present a modern re-interpretation of the linearised Laplace methodology described in the previous section. The key observation, made by Khan et al. (2019b), is that the GGN-Laplace posterior matches the true posterior of the tangent linear model h. Using a similar reasoning, Immer et al. (2021b) argue that the GGN-Laplace posterior should be paired with the tangent linear model at prediction time. The authors show that this results in more accurate posterior predictive distributions, which we reproduce in Figure 3.5. Building on this, Antorán et al. (2022) and Antorán et al. (2023) present a linearisation-first derivation of linearised Laplace, which we go on to present here.

The linearised Laplace method consists of two consecutive approximations, the latter of which is necessary only if ℓ or $\mathcal{R}$ are non-quadratic. That is, if the likelihood or prior are non-Gaussian.

1. We take a first-order Taylor expansion of g around $v_\star$, yielding the surrogate model given in (3.33). This model's weights linearly combine the rows of the Jacobian matrix, which can be seen as a feature expansion of the input. To make this connection explicit, we henceforth adopt the notation $\phi(x) = J(x)$. The linear model's loss is

$$\mathcal{L}_h(w) = \sum_{i=1}^{n} \ell(y_i, h(w, x_i)) + \mathcal{R}(w). \tag{3.37}$$

   If this expression is quadratic, we may proceed with conjugate linear-Gaussian inference as described in Chapter 2. The Laplace approximation is not needed.

2. If the linear model's loss is non quadratic, we locally approximate it with the Laplace method. This yields a Gaussian posterior of the form

$$\mathcal{N}(v_\star,\ (\partial_v^2 \mathcal{L}_h(v_\star))^{-1}). \tag{3.38}$$

   Since the NN and tangent linear model share gradients, that is $\partial_v \mathcal{L}_g(v) = \partial_h \mathcal{L}_h(w)$, if $v_\star$ is a local optima of $\mathcal{L}_g$ it will also be one of $\mathcal{L}_h$. Direct calculation shows that $\partial_v^2 \mathcal{L}_h(v_\star) = A + \Phi^T B \Phi = H$, for $\nabla_w^2 \mathcal{R}(v_\star) = A$ and B a block diagonal matrix with

blocks $B_i = \nabla^2_{\hat{y}_i} \ell(y_i, \hat{y}_i)$ evaluated at $\hat{y}_i = h(v_\star, x_i) = g(v_\star, x_i)$. We have once again used notation matching the one used for conjugate Gaussian-linear models in Chapter 2 to highlight the Laplace approximation's Gaussianisation of the likelihood and prior.

Linearised Laplace has returned us a conjugate Gaussian multi-output linear model with the GGN as its posterior precision.

Remark Linearisation as a modelling choice
We could go a step further and view the linearisation step as a modelling choice. That is, we would be adopting a Gaussian linear model with basis functions matching the NN's Jacobian around $v_\star$. This would not be a data independent basis, however. The NN has been fit to our dataset in order to find $v_\star$. We overlay the Jacobian basis functions on top of a toy 1d dataset on which the corresponding NN was trained in Figure 3.6. The basis functions present sharp changes in the input regions where there is data, and are more smooth elsewhere. The same is true for the equivalent kernel. Given this data dependence, it is somewhat surprising that linearised Laplace does not result in uncertainty underestimation given that we are using our data twice: one to train our NN and another for inference in the tangent linear model. The most common case of double use of data resulting in overfitting is hyperparameter learning with the model evidence. This overfitting happens, for instance, in the deep kernel learning model (Ober and Rasmussen, 2019; Ober et al., 2021), which uses the- linear model evidence to fit basis functions parametrised by neural networks. My intuition is that linearised Laplace escapes overfitting because the NN's weights are not trained with the linearised model's evidence, but with some bespoke NN loss function. As a result, the Jacobian basis functions do not perfectly pass through the training targets.

The linearised posterior distribution over NN outputs at a new input $x' \in \mathcal{X}$ is thus

$$\mathcal{N}(g(v_\star, x'),\ \phi(x')H^{-1}\phi(x')^T), \tag{3.39}$$

matching the expression used by (MacKay, 1992b) and given in (3.34). In other words, linearised Laplace simply augments our pre-trained NN's predictions with with Gaussian errorbars. Keeping the NN outputs as the mean presents a large advantage over alternative approaches to Bayesian inference in deep learning which often trade off goodness of fit with quality of uncertainty estimates (Daxberger et al., 2021a,b; Snoek et al., 2019a). Additionally, linearised Laplace tends to provide sensibly shaped errorbars, contrasting with

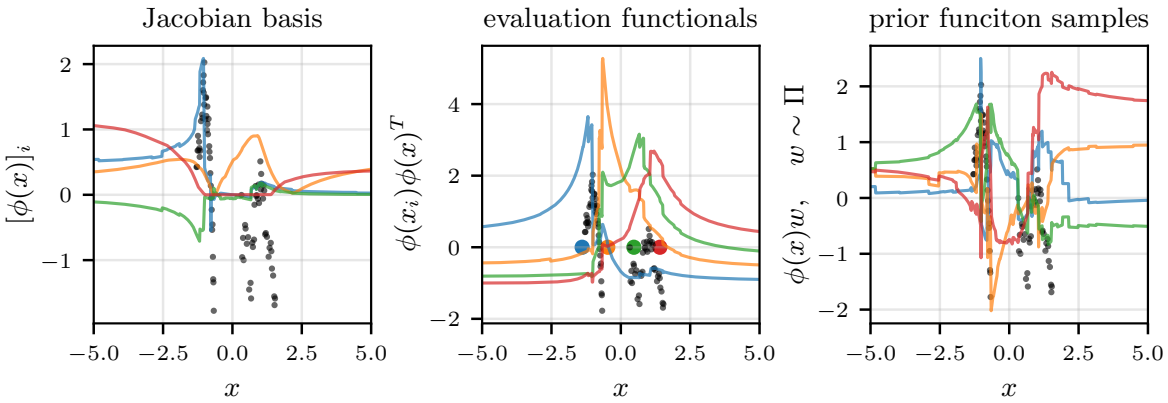


Fig. 3.6 Illustration of the prior implied by the linearised NN. The leftmost plot shows 4 dimensions of the Jacobian basis (i.e. the Jacobian with respect to 4 of the NN weights) function of a 2 layer residual MLP trained on the 1d toy dataset introduced by Antorán et al. (2020). This dataset is displayed as black dots. The middle plot shows the kernel implied by the Jacobian basis. It is non-stationary. The rightmost plot shows 4 samples drawn from the linearised NN prior, with the NN loss mode's prediction $g(v_\star, \cdot)$ removed.

other approximations which fail simple tests like "in-between" uncertainty (Foong et al., 2019b) or "far-away" uncertainty (Kristiadi et al., 2020).

Remark **Connections to the neural tangent kernel and infinitely wide NNs**
The Neural Tangent Kernel (NTK) (Jacot et al., 2018; Lee et al., 2019) is intimately related to linearised Laplace. The NTK matches the linearised model given in (3.33), but with the Taylor expansion point being the point where the NN weights are initialised, instead of an optima of the loss. As the NN width increases, and under some relatively weak conditions which we will not discuss here, the mode of the NN loss goes to the initialisation point. In this setting, the linearised Laplace posterior matches the posterior of a GP with the NTK as its covariance kernel. This distribution is different, however, from the true posterior of the infinitely wide NN model (de G. Matthews et al., 2018). The latter also corresponds to a GP, but its kernel is not the NTK. It is the outer product of the Jacobians of the NN's last layer weights. Thus, the NTK is a sum of the infinitely wide NN kernel and also some other kernels with features matching the NN's non-last layer Jacobians.

3.3.3 Learning hyperparameters with the Laplace evidence

An important limitation of linearised Laplace is its predictive variance's sensitivity to the curvature of the likelihood B and regulariser A. The values of these matrices derived from the loss used to train the NN often result in miss-calibrated uncertainty, and large performance gains can be obtained by tuning them. To this end, the Laplace approximation provides us with and an estimate of the model evidence, which may be used to learn A and B as well as other hyperparameters.

Again denoting our set of hyperparameters as θ, we re-arrange Bayes rule (2.8) to expose the model evidence on the left hand side, substitute the posterior density for our Gaussian approximation (3.38), and evaluate the functions at $v_\star$, obtaining

$$\begin{aligned}\log p(Y;\,\theta) &\approx \log p(Y|v_\star) + \log \pi(v_\star) - \mathcal{N}(v_\star; v_\star, H^{-1}) \\ &= \log p(Y|v_\star) + \log \pi(v_\star) - \frac{1}{2}\log \det H + \frac{d}{2}\log(2\pi) \coloneqq \mathcal{G}_{v_\star}(\theta),\end{aligned} \tag{3.40}$$

where we make the dependence of the approximation on the linearisation point $v_\star$ explicit by introducing it as a subscript to $\mathcal{G}$. If we do not have access to the explicit joint density of parameters and observations $p(Y|v_\star)\pi(v_\star)$, we may use the loss function $\mathcal{L}(v_\star)$ instead. However, we have to introduce the normalisation factors for the respective Gaussian approximations of the likelihood and prior

$$\mathcal{G}_{v_\star}(\theta) = -\mathcal{L}_f(v_\star) - \frac{1}{2}\log \det H + \frac{1}{2}\log \det A + \frac{1}{2}\log \det B - \frac{n}{2}\log(2\pi). \tag{3.41}$$

We may tune the hyperparameters θ for a NN by choosing them to maximise $\mathcal{G}_{v_\star}(\theta)$. This may improve our errorbar calibration[4], but will not change our NN's outputs however, as its parameters are held fixed at $v_\star$. Mackay (1992a) proposes to re-train the NN from scratch using the new hyperparameters. Steps of NN training and hyperparameter optimisation are iterated until a joint stationary point of the parameters and hyperparameters is found

$$v_\star \in \underset{v\in\mathbb{R}^d}{\arg\min}\, \mathcal{L}_f(v, \theta_\star) \quad \text{and} \quad \theta_\star \in \underset{\theta\in\Theta}{\arg\min}\, \mathcal{G}_{v^\star}(\theta) \tag{3.42}$$

where we have made the loss' dependence on the hyperparameters explicit by adding them as an argument.

[4]It most likely will not and fixing this is the object of Chapter 5. But one could plausibly conclude that it might from reading the relevant literature.

3.3.4 Online Laplace methods

The size of the neural networks and datasets has grown dramatically since 1992. As a result, nowadays, re-training our NN multiple times after hyperparameter updates introduces a prohibitive computational cost. This motivates Online Laplace (OL) approaches which, at timestep t with parameters v_t and hyperparameters θ_t, perform a step of NN parameter optimisation to minimise $\mathcal{L}_f(v_t; \theta_t)$, obtaining v_{t+1}, followed by a hyperparameter update to maximise $\mathcal{G}_{v_{t+1}}(\theta_t)$ (Foresee and Hagan, 1997; Friston et al., 2007; Immer et al., 2021a)[5]. Critically, the Laplace approximation of the evidence is constructed with both the NN loss and GGN evaluated at the current NN parameter setting v_{t+1}. Since optimisation has not converged, $v_{t+1} \notin \arg\min_{v \in \mathbb{R}^d} \mathcal{L}_f(v; \theta)$. Thus $\mathcal{G}_{v_{t+1}}(\theta_t)$, which discards the first-order Taylor expansion term, is unlikely to provide a local approximation to the true model evidence. Despite this, online Laplace methods have seen success recently, for instance for learning data augmentation hyperparameters (Immer et al., 2022) and model invariance hyperparameters (van der Ouderaa et al., 2023).

In Lin et al. (2023a), a piece of work not covered in this thesis, we construct a Taylor expansion-based Gaussian approximation to the evidence that does not discard the first order term, and thus may be more suitable for online use. We then show that this approximation corresponds to the exact evidence of the tangent linear model. Interestingly, when we drop the first order term, we recover a variational lower bound on the evidence of the linear model, providing some justification for the online approaches of Foresee and Hagan (1997); Friston et al. (2007); Immer et al. (2021a).

3.3.5 Limitations of the linearised Laplace approximation

Linearised Laplace presents a number of critical limitations and addressing these is the object of much of the rest of this thesis.

Linearised Laplace shares the limitations of linear model inference discussed in Chapter 2: cubic compute cost and quadratic memory cost, in either the number of NN parameters d or the number of outputs times observations nc. In modern deep learning problems, both of these quantities tend to be in the tens of millions, or larger. Fortunately, linearising the NN allows us to leverage the approximations for linear models discussed in Section 3.1, Section 3.2, and the ones we will introduce next, in Chapter 4. Additionally, there are a number of approximations that exploit the structure of the linearised NN, such as last layer methods

[5]Friston et al. (2007) refers to the described online Laplace procedure as *Variational Laplace*.

(Eschenhagen et al., 2021; Kristiadi et al., 2020), Kronecker factorised approximations (Immer et al., 2023b; Ritter et al., 2018), and subnetwork methods (Daxberger et al., 2021b).

Additionally, linearised Laplace presents some unique limitations. Firstly, its basis functions, the NN Jacobians, are computationally expensive to deal with. For a given in put $x \in \mathcal{X}$, computing the Jacobian expansion $J(x)$ requires either c passes of backward mode Automatic Differentiation (AD) or d passes of forward mode AD. Clearly the former is preferable for most neural networks, but it still presents an issue when dealing with high-dimensional output spaces, for instance, in many-way image classification, image-restoration, or language modelling. For a large enough model, even storing the $c \times d$ dimensional Jacobian features may be too expensive. In a textbook implementation of linearised Laplace, Jacobians appear at two different points: 1) when constructing the GGN matrix at inference time, and 2) when making predictions for a new observation x'. The former problem can be partially ameliorated by leveraging the equivalency between the GGN and the Fisher information matrix for exponential family likelihoods—almost all of the ones used in machine learning. In particular, the Fisher admits unbiased stochastic estimation by sub-sampling output dimensions (Kunstner et al., 2019; Martens, 2014). Unfortunately, the predictive covariance does not admit this sort of approximation. Chapter 6 presents an implementation of linearised Laplace that completely avoids instantiating Jacobian matrices.

Finally, astute readers may have noticed that the assumption that we find a local optima of the NN training loss is unrealistic. NN optimisation landscapes present a large number of symmetries and invariances, and stochastic optimisation is almost always used to minimise them. Furthermore, early stopping is also almost always used. These techniques prevent us from finding a local minima of the loss. This is intentional, it can be thought of as regularisation, because reaching a very low loss value would almost surely mean we are overfitting. One may thus wonder how not having access to a minimum of the loss affects the linearised Laplace approximation? This is addressed in Chapter 5.

Chapter 4

Sampling from Gaussian Process posteriors using Stochastic Gradient descent

"When solving a given problem, try to avoid solving a more general problem as an intermediate step." — Vladimir Vapnik

Gaussian processes (GPs) provide a comprehensive framework for learning unknown functions in an uncertainty-aware manner. This often makes GPs the model of choice for sequential decision-making, achieving state-of-the-art performance in tasks such as optimising molecules in computational chemistry settings (Gómez-Bombarelli et al., 2018) and automated hyperparameter tuning (Hernández-Lobato et al., 2014; Snoek et al., 2012).

As we have seen in previous chapters, the main limitation of Gaussian processes is that their computational cost is cubic in the training dataset size. Significant research efforts have been directed at addressing this limitation, resulting in two key classes of scalable inference methods: (i) *inducing point* methods (Hensman et al., 2013; Titsias, 2009a), which approximate the GP posterior, and (ii) *conjugate gradient* methods (Artemev et al., 2021; Gardner et al., 2018; Gibbs and MacKay, 1996), which approximate the computation needed to obtain the GP posterior. Note that in structured settings, such as geospatial learning in low dimensions, specialised techniques are available (Wilkinson", 2019; Wilson and Nickisch, 2015). Throughout this chapter, we focus on the generic setting, where scalability limitations are as of yet unresolved.

In recent years, stochastic gradient descent (SGD) has emerged as the leading technique for training deep learning models at scale (Ruder, 2016; Tian et al., 2023). It has also been applied to kernel methods (Dai et al., 2014), and even connected to variational Bayesian inference (Mandt et al., 2017). While the principles behind the effectiveness of SGD are not yet fully understood, empirically, SGD often leads to good predictive performance—even when it does not fully converge. The latter is the default regime in deep learning, and has motivated researchers to study *implicit biases* and related properties of SGD (Belkin et al., 2019; Zou et al., 2021).

In the context of GPs, SGD is commonly used to learn kernel hyperparameters—by optimising the marginal likelihood (Chen et al., 2020, 2022; Gardner et al., 2018) or closely related variational objectives (Hensman et al., 2013; Titsias, 2009a). In this chapter, we explore applying SGD to the complementary problem of approximating GP posterior samples given fixed kernel hyperparameters. In one of his seminal books on statistical learning theory, Vladimir Vapnik (1995) famously said: *"When solving a given problem, try to avoid solving a more general problem as an intermediate step."* Motivated by this viewpoint, as well as the aforementioned property of good performance often not requiring full convergence when using SGD, we ask: *Do the linear systems arising in GP computations necessarily need to be solved to a small error tolerance? If not, can SGD help accelerate these computations?*

We answer the latter question affirmatively, with specific contributions as follows. (i) In Section 4.2, we develop a scheme for drawing GP posterior samples by applying SGD to a quadratic problem. In particular, we re-cast the pathwise conditioning technique of (Wilson et al., 2020) as an optimisation problem, and, in Section 4.3, extend the method to inducing point GPs. In Section 4.2.2, we develop a novel low-variance SGD sampling estimator applicable to both linear models, where the kernel is finite dimensional, and GPs. For the kernelised setting, in Section 4.2.3, we introduce Stochastic Dual Descent (SDD), an optimisation scheme that targets a better conditioned dual objective in place of the more-common kernel ridge regression objective. (ii) In Section 4.4, we characterise the implicit bias in SGD-approximated GP posteriors showing that despite optimisation not fully converging, these match the true posterior in regions both near and far away from the data. (iii) Finally, in Section 4.5, we present the following experimental evidence:

1. On standard UCI regression benchmarks with up to 2 million observations, stochastic dual descent either matches or improves upon the performance of conjugate gradients, while strictly outperforming other baselines.

2. On large-scale parallel Bayesian optimisation, stochastic gradient descent is shown to be superior to preconditioned conjugate gradients and inducing point variational

inference, both in terms of the number of iterations and in terms of wall-clock time. In turn, stochastic dual descent is shown to be superior to vanilla stochastic gradient descent.

3. On a molecular binding affinity prediction task, where Gaussian processes have not previously been shown to be competitive with deep learning approaches, the performance of stochastic dual descent matches that of graph neural networks.

The methods and insights developed in this chapter, in particular the low-variance SGD estimator of weight-space posterior samples, will play a key role in scaling the linearised Laplace method to large scale neural networks and datasets in Chapter 6.

4.1 Pathwise conditioning as an optimisation problem

Both a Gaussian process' posterior mean and posterior samples can be expressed as solutions to quadratic optimisation problems. For the primal, weight-space form, the expressions for the mean and samples were provided in Chapter 2, in (3.37) and (2.38), respectively. Here we study the more general kernelised form. To simplify notation, we assume the output dimension is $c = 1$ throughout this chapter. As a result, our noise precision matrix B is diagonal. Additionally, we assume our kernel k is stationary, or at least admits random features.

The GP posterior mean minimises the ridge regression loss over functions in the RKHS:

$$f_\star(\cdot) = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} [B]_{ii} (y_i - \langle k(x_i, \cdot), f \rangle)^2 + \|f\|_{\mathcal{H}}^2. \tag{4.1}$$

Using the representer theorem (Schölkopf et al., 2001), we transform this objective into a quadratic problem over the representer weights $\alpha \in \mathbb{R}^n$

$$f_\star(\cdot) = K_{(\cdot)X}\alpha_\star = \sum_{i=1}^{n} \alpha_{\star i} k(x_i, \cdot) \quad \alpha_\star = \arg\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} [B]_{ii} (y_i - K_{x_i X}\alpha)^2 + \|\alpha\|_K^2. \tag{4.2}$$

Its optima is $\alpha_\star = (K + B^{-1})^{-1} Y$, matching (2.16). Recall that we refer to $k(x_i, \cdot)$ as the *evaluation functionals*, and we henceforth refer to $\|\alpha\|_K^2 = \alpha^T K \alpha$ as the *regulariser*. To construct respective optimisation problem for obtaining posterior samples, we part from the decomposed pathwise expression given in (2.29), which we repeat here for the reader's

convenience

$$(f|Y)(\cdot) = \underbrace{f_\star(\cdot)}_{\text{posterior mean}} + \underbrace{f(\cdot)}_{\text{prior sample}} - \underbrace{K_{(\cdot)X}(K + B^{-1})^{-1}(f(X) + \mathcal{E})}_{\text{uncertainty reduction term}} \tag{4.3}$$
$$\text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad f \sim \text{GP}(0, k).$$

The posterior mean can be obtained by solving (4.2). We approximate the prior function sample f using a sum of random Fourier features $\widetilde{f}$, as described in Section 2.3.1. Each posterior sample's uncertainty reduction term is parametrised by a set of representer weights. These are given by a linear solve against a noisy prior sample evaluated at the observed inputs $(K + B^{-1})^{-1}(\widetilde{f}(X) + \varepsilon)$. Thus, by analogy to (4.2), we can construct an optimisation objective targeting a sample's representer weights as

$$\arg\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} [B]_{ii} (\widetilde{f}(x_i) + \varepsilon_i - K_{x_i X}\alpha)^2 + \|\alpha\|_K^2 \tag{4.4}$$
$$\text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad \widetilde{f}(\cdot) = \phi_s(\cdot) w \quad w \sim \mathcal{N}(0, I_d) \quad s \sim \Omega,$$

where ε_i are the individual entries of $\mathcal{E} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n]^T$. We denote as ϕ_s a d dimensional random feature expansion. Unless specified otherwise, we assume a stationary kernel and use a cosine expansion with random frequencies drawn from our kernel's spectral density Ω.

Remark Dividing and conquering
We explicitly separate the posterior mean and 0-mean samples into two separate optimisation problems: (4.2) and (4.4). However, this need not be the case. We could shift the solution of the sampling objective by exactly the mean function by regressing onto $(\widetilde{f}(X) + \mathcal{E} + Y)$ instead of $(\widetilde{f}(X) + \mathcal{E})$. Unfortunately, when doing this, we find the target vector Y to often dominate the objective, resulting in a worse quality estimates of the GP posterior variance.

4.2 Stochastic estimators of the sampling objective

We now develop and analyse techniques for drawing samples from GP posteriors using stochastic gradient descent. We provide three different stochastic estimators. First a simple, general purpose one in Section 4.2.1. This objective will prove useful when dealing with inducing point GPs, where the innovations discussed next are not applicable. Then, in Section

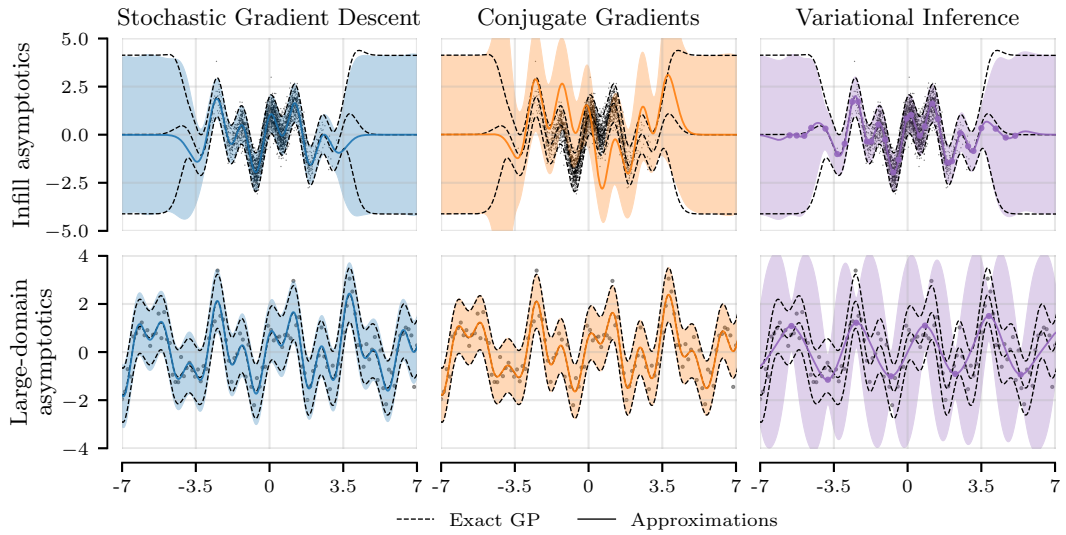


Fig. 4.1 Comparison of SGD, CG (Wang et al., 2019) and SVGP (Hensman et al., 2013) for GP inference with a squared exponential kernel on 10k datapoints from $\sin(2x) + \cos(5x)$ with observation noise distribution $\mathrm{N}(0, 0.5)$. We draw 2000 function samples with all methods by running them for 10 minutes on an RTX 2070 GPU. *Infill asymptotics* considers $x_i \sim \mathrm{N}(0, 1)$. A large number of points near zero result in a very ill-conditioned kernel matrix, preventing CG from converging. SGD converges in all of input space except at the edges of the data. SVGP can summarise the data with only 20 inducing points. Note that CG converges to the exact solution if one uses more compute, but produces significant errors if stopped too early, as occurs under the given compute budget. *Large domain asymptotics* considers data on a regular grid with fixed spacing. This problem is better conditioned, allowing SGD and CG to recover the exact solution. However, 1024 inducing points are not enough for SVGP to summarise the data.

4.2.2, one with reduced variance when drawing 0-mean posterior samples. We will also provide the weight-space counterpart of this estimator. We will go on to investigate the conditioning of the quadratic objectives targeted by these estimators. This will lead us to develop our third method *Stochastic Dual Descent* in Section 4.2.3, which brings favourable conditioning to the kernelised setting. Finally, Section 4.2.4 compares different approaches to stochastic optimisation and provides guidelines on best practices. As a preview of this section's contributions, we showcase SGD's performance, and compare it to CG and inducing point VI, on a pair of toy problems designed to capture complementary computational difficulties, in Figure 4.1.

4.2.1 A first approach: mini batching and unbiased random features

The optimisation problem (4.2), requires $\mathcal{O}(n^2)$ operations to compute both the square error and regulariser terms exactly. The square error loss term is amenable to minibatching, which gives an unbiased estimate in $\mathcal{O}(n)$ operations. Assuming that k admits random features, we can stochastically estimate the regulariser by expressing the kernel matrix as the expectation of an outer product of feature expansions (see Section 2.2.4). That is, $\|\alpha\|_K^2 = \mathbb{E}_{s\sim\Omega}\alpha^T\Phi_s\Phi_s^T\alpha$ where $\Phi_s \in \mathbb{R}^{n\times d}$ is the stacked d-dimensional random feature expansion of the n inputs. Combining both estimators gives our SGD objective

$$\frac{n}{r}\sum_{i=1}^{r}[B]_{ii}(y_i - K_{x_iX}\alpha)^2 + \alpha^T\Phi_s\Phi_s^T\alpha \tag{4.5}$$

where r is the minibatch size. This regulariser estimate is unbiased even when drawing a single Fourier feature per step $d = 1$. The number of features controls the variance. Evaluating (4.5) presents $\mathcal{O}(n)$ complexity, in contrast with the $\mathcal{O}(n^2)$ complexity of one CG step. It is straight forward to apply the same estimators to the 0-mean sampling objective in (4.4) obtaining

$$\frac{n}{r}\sum_{i=1}^{r}[B]_{ii}(\widetilde{f}(x_i) + \varepsilon_i - K_{x_iX}\alpha)^2 + \alpha^T\Phi_s\Phi_s^T\alpha, \tag{4.6}$$

with a per-step cost of $\mathcal{O}(ns)$, for s the number of posterior samples drawn. We discuss sublinear inducing point techniques further on, in Section 4.3.

4.2.2 A lower variance estimator for SGD-based sampling

Empirically, the minibatch estimator in (4.6) results in high gradient variance. This is because our targets contain unstructured noise ε_i, which is difficult to predict. We propose an alternative sampling objective function which shares the same gradient in expectation, but whose stochastic estimates may present lower variance. We provide both kernelised and weight-space forms for the new objective. We then analyse the variance of the new weight-space objective.

Kernelised form

We modify the sampling objective (4.4) by moving the noise into the regulariser term

$$\arg\min_{\alpha\in\mathbb{R}^n} \sum_{i=1}^{n} [B]_{ii}(\widetilde{f}(x_i) - K_{x_i X}\alpha)^2 + \|\alpha - \mathcal{E}'\|_K^2 \tag{4.7}$$
$$\text{with} \quad \mathcal{E}' \sim \mathcal{N}(0, B) \quad \text{and} \quad \widetilde{f}(\cdot) = \phi_s(\cdot)w \quad w \sim \mathcal{N}(0, I_d) \quad s \sim \Omega,$$

which inverts the covariance of the distribution the noise is sampled from. We highlight this change with the prime notation $\mathcal{E}'$. This modification *preserves the optimal representer weights* since objective (4.7) equals (4.4) up to a constant.

Derivation Equivalency of kernelised sampling objectives
To show the equality of both objectives up to a constant, we show both have the same gradient.
Let $LL^T = B^{-1}$ be the Cholesky factorisation of the noise covariance, let $f(X) \sim \mathcal{N}(0, K)$, and let $\epsilon \sim \mathcal{N}(0, I_n)$. Our objectives are

$$\|f(X) + L\epsilon - K\alpha\|_B^2 + \|\alpha\|_K^2 \tag{4.8}$$

and

$$\|f(X) - K\alpha\|_B^2 + \left\|\alpha - L^{-T}\epsilon\right\|_K^2. \tag{4.9}$$

Taking derivatives with respect to α, we have

$$\partial\alpha \left(\|f(X) + L\epsilon - K\alpha\|_B^2 + \|\alpha\|_K^2\right) \tag{4.10}$$
$$= -2KB\left(f(X) + L\epsilon - K\alpha\right) + 2K\alpha \tag{4.11}$$
$$= -2K(Bf(X) - BK\alpha + L^{-T}\epsilon - \alpha), \tag{4.12}$$

and

$$\partial\alpha \left(\|f(X) - K\alpha\|_B^2 + \left\|\alpha - L^{-T}\epsilon\right\|_K^2\right) \tag{4.13}$$
$$= -2KB\left(f(X) - K\alpha\right) + 2K(\alpha - L^{-T}\epsilon) \tag{4.14}$$
$$= -2K(Bf(X) - BK\alpha + L^{-T}\epsilon - \alpha), \tag{4.15}$$

respectively. These expressions match, giving the claim. Furthermore, since both objectives are strictly convex, they both have the same unique minimum.

Weight-space form

We now apply the same trick for the weight-space form of the sample-then-optimise objective (2.38). This will allow us to scale the linearised Laplace method to real-world sized deep learning problems in Chapter 6. We begin by stating the zero-mean sample-then-optimise objective:

$$L(w) = \frac{1}{2}\|\mathcal{E} - \Phi w\|_B^2 + \frac{1}{2}\|w - w_0\|_A^2 \tag{4.16}$$
$$\text{with} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}).$$

Inspecting the expression, we consider how it may be stochastically estimated:

- The first term is data dependent. It corresponds to the scaled squared error in fitting $\mathcal{E}$ as a linear combination of Φ. Its gradient requires stochastic approximation for large datasets.
- The second term, a regulariser centred at w_0, does not depend on the data. Its gradient can thus be computed exactly at every optimisation step. This differs from the kernelised setting, where the regulariser contained the kernel matrix and required stochastic estimation.

Again, we encounter random noise in the targets, and thus, the variance of a mini-batch estimate of the gradient of $\|\Phi z - \mathcal{E}\|_B^2$ may be large. Instead, for $\mathcal{E}$ and w_0 defined as above, we propose the following alternative loss, again equal to (4.16) up to an additive constant independent of the variable being optimised:

$$L'(w) = \frac{1}{2}\|\Phi w\|_B^2 + \frac{1}{2}\|w - w_0'\|_A^2 \quad \text{with} \quad w_0' = w_0 + A^{-1}\Phi^T B\mathcal{E} \tag{4.17}$$
$$\text{where} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}).$$

The mini-batch gradients of L' and L are equal in expectation and both objective's optima is the same. However, in L', the randomness from the noise samples $\mathcal{E}$ and the prior sample w_0 both feature within the regularisation term—the gradient of which can be computed exactly—rather than in the data-dependent term.

Derivation Equivalency of weight space sampling objectives
Again, the losses L and L' are strictly convex, thus to confirm they have the same unique minimum, it suffices to consider the respective first order optimality conditions. We introduce $\zeta \in \mathbb{R}^d$ and $\zeta' \in \mathbb{R}^d$ such that $\partial_w L(\zeta) = 0$ and $\partial_w L'(\zeta') = 0$. We have,

$$\partial_w L(\zeta) = \Phi^T B(\Phi\zeta - \mathcal{E}) + A(\zeta - w_0), \tag{4.18}$$

and

$$\begin{aligned} \partial_w L'(\zeta') &= \Phi^T B\Phi\zeta' + A(\zeta' - A^{-1}\Phi^T B\mathcal{E} - w_0) && (4.19)\\ &= \Phi^T B(\Phi\zeta' - \mathcal{E}) + A(\zeta' - w_0) && (4.20) \end{aligned}$$

Thus $\zeta = \zeta'$ almost surely. Moreover, $L'(w) = L(w) + C$ for all w, for C a constant independent of w.

For completeness, we provide an alternative path to checking the validity of our sampling objectives. We study the distribution of their optima.

Derivation Distribution of optima of weight-space sampling losses
To determine the distribution of $\zeta = \arg\min_{w\in\mathbb{R}^d} L(w)$, we note that it is a linear transformation of zero-mean Gaussian random variables, and thus itself a zero-mean Gaussian random variable. Rearranging the first order optimality condition, we find that

$$\zeta = H^{-1}(\Phi^T B\mathcal{E} + A\theta^0). \tag{4.21}$$

Thus

$$\begin{aligned} \mathbb{E}[\zeta\zeta^T] &= H^{-1}\mathbb{E}[(\Phi^T B\mathcal{E} + A\theta^0)(\Phi^T B\mathcal{E} + A\theta^0)^T]H^{-1} && (4.22)\\ &= H^{-1}\left(\Phi^T B\mathbb{E}[\mathcal{E}\mathcal{E}^T]B\Phi + A\mathbb{E}[\theta^0\theta^0]A + 2\Phi^T B\mathbb{E}[\mathcal{E}(\theta^0)^T]A\right)H^{-1} && (4.23)\\ &= H^{-1}(\Phi^T B\Phi + A)H^{-1} = H^{-1}HH^{-1} = H^{-1}, && (4.24) \end{aligned}$$

and so $\zeta \sim \mathcal{N}(0, H^{-1})$.

Analysis of minibatch gradient variance

Consider the variance of the single-datapoint stochastic gradient estimators for both weight-space objectives' data dependent terms. At $z \in \mathbb{R}^d$, for datapoint indices sampled as $j \sim \text{Uniform}(\{1, \ldots, n\})$, these are

$$\hat{g} = n\phi(x_j)^T(\phi(x_j)z - \varepsilon_j) \quad \text{and} \quad \hat{g}' = n\phi(x_j)^T\phi(x_j)z \tag{4.25}$$

for L and L', respectively. Direct calculation, shows that

$$\tfrac{1}{n}\left[\text{Var}\hat{g} - \text{Var}\hat{g}'\right] = \text{Var}(\Phi^T B\mathcal{E}) - 2\text{Cov}(\Phi^T B\Phi z, \Phi^T B\mathcal{E}) =: \Delta. \tag{4.26}$$

Note that both $\text{Var}\hat{g}$ and $\text{Var}\hat{g}'$ are $d \times d$ matrices. We impose an order on these by considering their traces: we prefer the new gradient estimator $\hat{g}'$ if the sum of its per-dimension variances is lower than that of $\hat{g}$; that is if $\text{Tr}\,\Delta > 0$. We analyse two key settings:

- At initialisation, taking $w = w_0$ (or any other initialisation independent of $\mathcal{E}$),

$$\text{Tr}\,\Delta = \text{Tr}\{\Phi^T B\mathbb{E}[\mathcal{E}\mathcal{E}^T]B\Phi\} - \text{Tr}\{\Phi^T B\Phi\mathbb{E}[w_0\mathcal{E}^T]B\Phi\} = \text{Tr}\,M > 0. \tag{4.27}$$

  Recall that $M = \Phi^T B\Phi$. We used that $\mathbb{E}[\mathcal{E}\mathcal{E}^T] = B^{-1}$ and since $\mathcal{E}$ is zero mean and independent of w_0, we have $\mathbb{E}[w_0\mathcal{E}^T] = \mathbb{E}w_0\mathbb{E}\mathcal{E}^T = 0$. Thus, the new objective L' is always preferred at initialisation.

- At convergence, that is, at $\zeta = \arg\min_{w\in\mathbb{R}^d} L(w)$, assuming a prior precision of the form $A = aI$, a more involved calculation, contained in Appendix C.3 of Antorán et al. (2023), shows that L' is preferred if

$$2a\gamma > \text{Tr}\,M, \tag{4.28}$$

  where γ is the effective dimension (2.46). This is satisfied if the regulariser a is large relative to the eigenvalues of M, (see Appendix C.4 of Antorán et al. (2023)), that is, when the effective dimension is low and the parameters are not strongly determined by the data relative to the prior. In practise, we find this to be the case for most heavily overparametrised models, like linearised neural networks, which are central to the following chapters of this thesis.

When L' is preferred both at initialisation and at convergence, we expect it to have lower variance for most minibatches throughout training. Even if the proposed objective L' is not

preferred at convergence, it may still be preferred for most of the optimisation, before the noise is fit well enough.

Remark Sticking the landing... or not.
The regular sample-then-optimise objective (4.16) uses random noise as targets in its fit term. There may be an optima of this fit term where we can perfectly interpolate the noise targets. At this point, not only is the gradient of the fit term 0, but so is any minibatch estimate we construct. Of course, the regulariser prevents the optima of the full objective from matching the optima of the fit term.
On the other hand, the fit term of our proposed objective (4.17) has no targets (or 0 targets). Clearly, the regulariser prevents the optima of the proposed objective being the zero vector. Thus, at the optima, the data-fit term's gradient variance will not go to zero.
With this, we can build intuition for the result in (4.28). The weaker the regulariser, the closer the full objective optima is to the optima of the fit term, where the regular sample-then-optimise objective (4.16) presents lower variance.

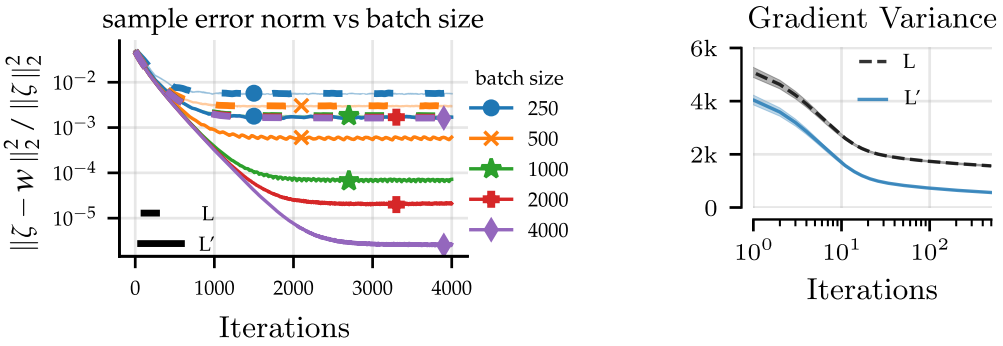


Fig. 4.2 Left: optimisation traces for the relative L_2 error in the weight-space posterior sample using our proposed sample-then-optimise objective L' (4.17) and the existing one L (4.16). The model is a linearised NN Section 7.2.1 and the task is MNIST. The plotted lines are averaged across 16 samples and 5 seeds. The low variance objective allows a $\approx 16\times$ reduction in batch size without reduction in weight-space posterior sample accuracy. Right: gradient variance throughout optimisation for a single-sample minibatch estimator ($r = 1$) of the kernelised sampling objectives. We use an RBF kernel on the ELEVATORS dataset ($n \approx 16$k). Again L' refers to the low variance estimator (4.7). In both plots we run SGD with Nesterov momentum $\rho = 0.9$ and geometric averaging.

Demonstration of low-variance sampling objective

Figure 4.2 illustrates the benefits of both our weight space and kernelised low variance sampling objectives. For the weight-space version, we use the Jacobian feature expansion corresponding to a LeNet style CNN with $d = 29226$ weights. Its linearisation point is found by pre-training the model on MNIST.

4.2.3 Stochastic Dual Descent

We now analyse the curvature of the quadratic objectives used for GP posterior sampling in Section 4.2.1 and Section 4.2.2. This leads us propose a better conditioned objective. In the context of this new objective, we question our previous choice of stochastic approximation. We compare mini-batching and random-feature approximations, and building upon the insights gained, propose a random-coordinate estimator with more desirable properties than either. The resulting algorithm: Stochastic dual descent (SDD) can be seen as an adaptation of the stochastic dual coordinate ascent algorithm of Shalev-Shwartz and Zhang (2013) to the large-scale deep-learning-type gradient descent framework. We also incorporate insights on stochastic approximation from the theoretical work of Dieuleveut et al. (2017) and Varre et al. (2021).

Assuming an isotropic noise precision, $B = bI$, the kernelised posterior sampling objectives provided in Section 4.2.1 and Section 4.2.2 are of the form

$$L_p(\alpha) \coloneqq \frac{1}{2}\|z - K\alpha\|^2 + \frac{b^{-1}}{2}\|\alpha\|_K^2$$

over $\alpha \in \mathbb{R}^n$ and for some choice of target vector $z \in \mathbb{R}^n$. In the kernel literature (Shalev-Shwartz and Zhang, 2013; Smola and Schölkopf, 1998), the kernel ridge regression objective L_p is known as the *primal* objective, and thus the subscript $_p$. We adopt this naming in the context of this subsection[1]. The *primal* gradient and Hessian are

$$\partial_\alpha L_p(\alpha) = K(b^{-1}\alpha - z + K\alpha) \quad \text{and} \quad \partial_\alpha^2 L_p(\alpha) = K(K + b^{-1}I), \tag{4.29}$$

respectively. Recall that the speed at which our optimiser approaches $\alpha_\star = (K + b^{-1}I)^{-1}z \in \mathbb{R}^n$ is determined by the condition number of the Hessian: the larger the condition number, the slower the convergence speed. The intuitive reason for this correspondence is that, to

[1]In the Bayesian linear model and Gaussian process literature, the weight space view is refereed as the *primal form*, while the kernelised view is refereed to as the *dual form*. In the kernel literature, the opposite is true; methods that deal with objects living in the RKHS are referred to as dual.

guarantee convergence, the step-size needs to scale inversely with the largest eigenvalue of the Hessian, while progress in the direction of an eigenvector underlying an eigenvalue is governed by the step-size multiplied with the corresponding eigenvalue. Letting $\lambda_i : \mathbb{R}^{n\times n} \to \mathbb{R}$ return the ith largest eigenvalue of a matrix, for the primal objective, the tight bounds on the relevant eigenvalues are

$$0 \leq \lambda_n(K(K + b^{-1}I)) \leq \lambda_1(K(K + b^{-1}I)) \leq \kappa n(\kappa n + b^{-1}),$$

where $\kappa = \sup_{x\in\mathcal{X}} k(x, x)$ is finite by assumption. These bounds only allow for a step-size β on the order of $(\kappa n(\kappa n + b^{-1}))^{-1}$ and, since they do not bound the minimum eigenvalue away from zero, we do not have a priori guarantees for the performance of gradient descent.

A dual objective

Consider, instead, minimising the *dual objective*

$$L_d(\alpha) = \frac{1}{2}\|\alpha\|^2_{K+b^{-1}I} - \alpha^T z. \tag{4.30}$$

The dual L_d has the same unique minimiser as L_p, namely $\alpha_\star$. We go on to show the duality of $(L_p, b^{-1}L_d)$; the factor of b^{-1} is immaterial.

Derivation Strong duality of SDD objective L_d (4.30)
We claim that

$$\min_{\alpha\in\mathbb{R}^n} L_p(\alpha) = -b^{-1} \min_{\alpha\in\mathbb{R}^n} L_d(\alpha),$$

with $\alpha_\star$ minimising both L_p and L_d.
That $\alpha_\star$ minimises both L_p and L_d can be established from the first order optimality conditions. Now, for the duality, observe that we can write $\min_{\alpha\in\mathbb{R}^n} L_p(\alpha)$ equivalently as the constrained optimisation problem

$$\min_{u\in\mathbb{R}^n} \min_{\alpha\in\mathbb{R}^n} \frac{1}{2}\|u\|^2 + \frac{b^{-1}}{2}\|\alpha\|^2_K \quad \text{subject to} \quad u = K\alpha - z\,.$$

Observe that this is quadratic in both u and α. Introducing Lagrange multipliers $\eta \in \mathbb{R}^n$, in the form $b^{-1}\eta$, where we recall that $b^{-1} > 0$, the solution of the above is equal to

that of

$$\min_{u\in\mathbb{R}^n}\min_{\alpha\in\mathbb{R}^n}\sup_{\eta\in\mathbb{R}^n}\frac{1}{2}\|u\|^2+\frac{b^{-1}}{2}\|\alpha\|_K^2+b^{-1}\eta^T(z-K\alpha-u)\,.$$

This is a finite-dimensional quadratic problem, and thus we have strong duality (see, e.g., Examples 5.2.4 in Boyd and Vandenberghe, 2014). We can therefore exchange the order of the minimum operators and the supremum, yielding the again equivalent problem

$$\sup_{\eta\in\mathbb{R}^n}\left\{\min_{u\in\mathbb{R}^n}\frac{1}{2}\|u\|^2-b^{-1}\eta^Tu\right\}+\left\{\min_{\alpha\in\mathbb{R}^n}\frac{b^{-1}}{2}\|\alpha\|_K^2-b^{-1}\eta^TK\alpha\right\}+b^{-1}\eta^Tz.$$

Noting that the two inner minimisation problems are quadratic, we solve these analytically using the first order optimality conditions, that is $\alpha=\eta$ and $u=b^{-1}\eta$, to obtain that the above is equivalent to

$$\sup_{\eta\in\mathbb{R}^n}-b^{-1}\left(\frac{1}{2}\|\eta\|_{K+b^{-1}I}^2-\eta^Tz\right)=-b^{-1}\min_{\eta\in\mathbb{R}^n}L_d(\eta)\,.$$

The result follows by chaining the above equalities.

The *dual gradient* and Hessian are given by

$$\partial_\alpha L_d(\alpha)=b^{-1}\alpha-z+K\alpha\quad\text{and}\quad\partial_\alpha^2L_d(\alpha)=K+b^{-1}I. \tag{4.31}$$

Observe that when running gradient descent on the dual objective L_d, we can use a step-size of order $(\kappa n+b^{-1})^{-1}$. That is, κn higher than before. Moreover, since the condition number of the dual satisfies $\mathrm{cond}(K+b^{-1}I)\leq 1+\kappa nb$, we have faster convergence, and can provide an a priori bound on the number of iterations required for any fixed error level for any length n sequence of observations.

Remark The conditioning of weight-space sampling objectives
The weight space sampling objectives (4.16) and (4.17) both present the Hessian $M+A=H$. Here, the Hessian of the data fit term appears only once, instead of twice (like in the Hessian of L_p (4.29)). For $A=aI$, $a\leq\lambda_d(H)$, and thus the conditioning number can be bounded from above. Thus, the improvements from the dual kernelised objective are already baked-into non-kernelised weight-space objectives. This makes

sense, since in Section 2.2.1 we saw that the weight space problem is also a dual to the kernelised regression problem (but a different dual than L_d).

Remark An RKHS view of the dual gradient
In Section 4.1, we derived the primal objective L_p (4.29) by applying the representer theorem to the regularised regression problem formulated in the RKHS

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - \langle k(x_i, \cdot), f\rangle^2 + \frac{1}{2}b^{-1}\|f\|^2_{\mathcal{H}}.$$

We did this because we can not fit infinite dimensional objects into our computers. However, an alternative could have been to take gradients directly in the RKHS

$$f' = K_{(\cdot)X}(Y - f(X)) + b^{-1}f,$$

apply them to get our updated function

$$f_{\text{new}} = f - \beta f' = f - \beta(K_{(\cdot)X}(Y - f(X)) + b^{-1}f),$$

and then express this in terms of representer weights

$$f_{\text{new}} = f - \beta f' = K_{(\cdot)X}\left(\alpha - \beta(Y - f(X) + b^{-1}\alpha)\right),$$

and thus we have

$$\alpha_{\text{new}} = \alpha - \beta(Y - f(X) + b^{-1}\alpha) = \alpha - \beta\partial_\alpha L_d(\alpha).$$

In other words, we can derive the dual objective by performing gradient descent in the RKHS and projecting back onto the representer weights once the gradient update has been performed. Thus, the dual objective is "dual" in the sense that it operates directly in the RKHS.

Demonstration: dual gradients

To illustrate the discussion so far, we compare the progress of dual and the primal gradient descent when computing the GP posterior mean representer weights on the UCI POL regression

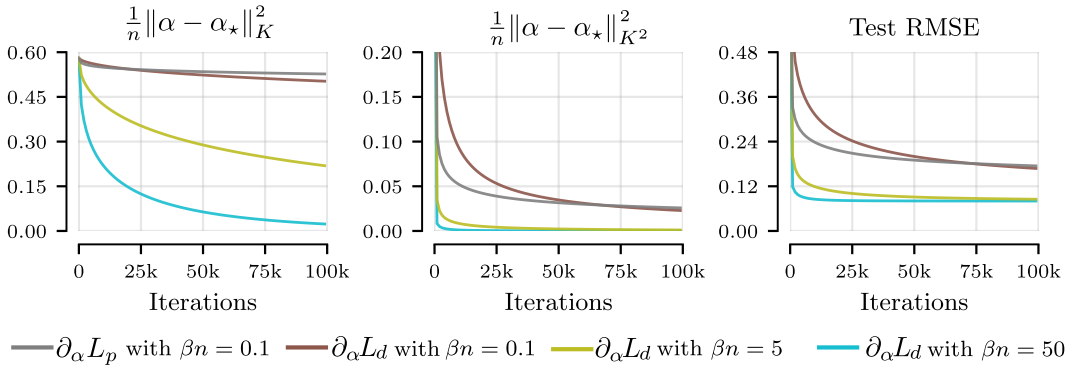


Fig. 4.3 Comparison of full-batch primal and dual gradient descent on POL with varying step-sizes. Primal gradient descent becomes unstable and diverges for βn greater than 0.1. Dual gradient descent is stable with larger step-sizes, allowing for markedly faster convergence than the primal. For $\beta n = 0.1$, the dual method makes more progress in the K-norm, whereas the primal in the K^2-norm.

task, with results shown in Figure 4.3. There, for the step-sizes we tried, gradient descent with the primal objective was only stable up to $\beta n = 0.1$, and diverged for larger step-sizes. In contrast, gradient descent with the dual objective is stable with a step-size as much as $500\times$ higher. It converges faster and to a better solution. We show this on three evaluation metrics: 1) distance to $\alpha^\star$ measured in $\|\cdot\|_K^2$, the K-norm (squared), 2) in $\|\cdot\|_{K^2}^2$, the K^2-norm (squared), and 3) test set root mean square error (RMSE). To understand the difference between the two norms, note that the K-norm error bounds the error of approximating $f_\star = K_{(\cdot)X}\alpha_\star$ with $f = K_{(\cdot)X}\alpha$ uniformly. Indeed, as shown below, we have the bound

$$\|f - f_\star\|_\infty \leq \sqrt{\kappa}\|\alpha - \alpha_\star\|_K^2\,, \tag{4.32}$$

where $\kappa = \sup_{x\in\mathcal{X}} k(x,x)$. Uniform norm guarantees of this type are crucial for sequential decision making tasks, such as Bayesian optimisation, where test input locations may be arbitrary. The K^2-norm metric, on the other hand, reflects training error. Examining the gradients, it is immediate that the primal gradient optimises for the K^2-norm, while the dual for the K-norm. And indeed, we see in Figure 4.3 that when both methods use $\beta n = 0.1$, up to 70k iterations, the dual method is better on the K-norm metric and the primal on K^2. Later, the dual gradient method performs better on all metrics. This, too, is to be expected, as the minimum eigenvalue of the Hessian of the dual loss is higher than that of the primal loss.

Derivation Uniform bound on function approximation error
We first introduce some machinery that allows us to formally reason about elements in a (potentially) infinite dimensional RKHS $\mathcal{H}$. For observations X, let $K_{X(\cdot)}\colon \mathcal{H} \to \mathbb{R}^n$ be the linear operator mapping $h \mapsto f(X)$, where $f(X) = (f(x_1), \ldots, f(x_n))$. We will write $K_{(\cdot)X}$ for the adjoint of $K_{X(\cdot)}$, and observe that K is the matrix of the operator $K_{X(\cdot)}K_{(\cdot)X}$ with respect to the standard basis.
With that, first, observe that,

$$\begin{aligned}
\|f - f_\star\|_\infty &= \sup_{x\in\mathcal{X}} |f(x) - f_\star(x)| && \text{(defn. of sup norm)}\\
&= \sup_{x\in\mathcal{X}} |\langle k(x,\cdot), f - f_\star\rangle| && \text{(reproducing property)}\\
&\leq \sup_{x\in\mathcal{X}} \|k(x,\cdot)\|_{\mathcal{H}}\|f - f_\star\|_{\mathcal{H}} && \text{(CBS)}\\
&\leq \sqrt{\kappa}\|f - f_\star\|_{\mathcal{H}}\,, && \text{(defn. of } \kappa)
\end{aligned}$$

Now, observe that $f = K_{(\cdot)X}\alpha$ and $f_\star = K_{(\cdot)X}\alpha_\star$, and so we have the equalities

$$\begin{aligned}
\|f - f_\star\|_{\mathcal{H}}^2 &= \langle K_{(\cdot)X}(\alpha - \alpha_\star), K_{(\cdot)X}(\alpha - \alpha_\star)\rangle && \text{(defn. norm)}\\
&= \langle \alpha - \alpha_\star, K_{X(\cdot)}K_{(\cdot)X}(\alpha - \alpha_\star)\rangle && \text{(defn. adjoint)}\\
&= \|\alpha - \alpha_\star\|_K^2\,. && \text{(defn. } K)
\end{aligned}$$

Combining the above two equalities yields the claim.

Randomised Gradients: Random Features versus Random Coordinates

We now study the construction of a stochastic objective to estimate the dual gradient (4.31) in linear time. The minibatching plus random feature estimator presented in Section 4.2.1 is not suitable for the dual objective because the kernel matrix does not appear in the regulariser of the dual objective. However, it does appear in the data fit term. Thus, we compare random feature and minibatch estimators.

We begin with random features. Recall that $K = E_{s\sim\Omega}\Phi_s\Phi_s^T$ where $\Phi_s \in \mathbb{R}^{n\times d}$ is a d dimensional random feature expansion of X. It follows that

$$\widetilde{\partial}_\alpha L_d(\alpha) = b^{-1}\alpha - z + \Phi_s\Phi_s^T\alpha$$

gives an unbiased estimate of $\partial_\alpha L_d(\alpha)$.

An alternative is to use minibatching

$$\widehat{\partial}_\alpha L_d(\alpha) = n e_i e_i^T \partial_\alpha L_d(\alpha) = n e_i (b^{-1}\alpha_i - z_i + [K]_i \alpha) \quad \text{with} \quad i \sim \text{Uniform}\{1, \dots, n\}, \tag{4.33}$$

where e_i are the elements of the canonical basis, i.e. $e_1 = [1, 0, 0, \dots]^T$. Thanks to $\mathbb{E}[n e_i e_i^T] = I$, this is also an unbiased estimate of $\partial_\alpha L_d(\alpha)$. Note that the cost of calculating either $\widetilde{\partial}_\alpha L_d(\alpha)$ or $\widehat{\partial}_\alpha L_d(\alpha)$ is linear in n, achieving our goal of reduced computation time. Also, note that while $\widetilde{\partial}_\alpha L_d(\alpha)$, which we call the random feature estimate, is generally a dense vector, $\widehat{\partial}_\alpha L_d(\alpha)$, is sparse. Since all but one coordinates of $\widehat{\partial}_\alpha L_d(\alpha)$ are zero, we refer to this as the *random coordinate estimate*[2].

The nature of the noise introduced by these estimators, and thus their qualities, are quite different. In particular, one can show that

$$\left\| \widehat{\partial}_\alpha L_d(\alpha) - \partial_\alpha L_d(\alpha) \right\| \leq \left\| (n e_i e_i^T - I)(K + b^{-1} I) \right\| \|\alpha - \alpha_\star\| .$$

As such, the noise introduced by $\widehat{\partial}_\alpha L_d(\alpha)$ is proportional to the distance between the current iterate α, and the optima $\alpha^\star$. The noise goes to 0 when the optima is reached. This estimator does stick the landing! For $\widetilde{\partial}_\alpha L_d(\alpha)$, letting $\widetilde{K} = \Phi_s \Phi_s^T$, we have

$$\left\| \widetilde{\partial}_\alpha L_d(\alpha) - \partial_\alpha L_d(\alpha) \right\| = \left\| (\widetilde{K} - K)\alpha \right\| .$$

As such, the error in $\widetilde{\partial}_\alpha L_d(\alpha)$ is *not reduced* as α approaches $\alpha_\star$. In the optimisation literature, $\widetilde{\partial}_\alpha$ would be classed as an *additive noise* gradient oracle, whereas $\widehat{\partial}_\alpha$ as a *multiplicative noise* oracle (Dieuleveut et al., 2017). Intuitively, multiplicative noise oracles automatically reduce the amount of noise injected as the iterates get closer to their target. While harder to analyse, multiplicative noise oracles often yield better performance: see, for example, Varre et al. (2021).

Remark Incompatibility of multiplicative noise with the weight-space problem Unfortunately, the random coordinate estimator (4.33) is not applicable to the weight space sample-then-optimise objective (4.17). The multiplicative noise estimator relies on the data-fit and regulariser terms being minibatched jointly. This is possible in the kernelised setting, since they are both n dimensional. However, in the weight-space setting, the data-fit objective is composed of n terms, while the regulariser is composed

[2] In practise, we implement this estimator by sampling multiple coordinates at each step, not just one.

Algorithm 1: *Stochastic dual descent* for approximating $\alpha^\star = (K + b^{-1}I)^{-1}z$

Inputs: Kernel matrix K with rows $K_1, \ldots, K_n \in \mathbb{R}^n$, targets $z \in \mathbb{R}^n$,
likelihood precision $b > 0$, number of steps $T \in \mathcal{N}^+$, batch size $r \in \{1, \ldots, n\}$,
step size $\beta > 0$, momentum parameter $\rho \in [0, 1)$, averaging parameter $\chi \in (0, 1]$
Set $v_0 = 0$; $\alpha_0 = 0$; $\overline{\alpha}_0 = 0$; `// all in` $\mathbb{R}^n$
while $t \in \{1, \ldots, T\}$ **do**
  Sample $\mathcal{I}_t = (i_1^t, \ldots, i_r^t) \sim \text{Uniform}\{1, \ldots, n\}$ independently ; `// rand. coord.`
  $g_t = \frac{n}{r} \sum_{i \in \mathcal{I}_t} ((K_i + b^{-1}e_i)^T(\alpha_{t-1} + \rho v_{t-1}) - b_i)e_i$; `// gradient estimate`
  $v_t = \rho v_{t-1} - \beta g_t$; `// velocity update`
  $\alpha_t = \alpha_{t-1} + v_t$; `// parameter update`
  $\overline{\alpha}_t = \chi\alpha_t + (1 - \chi)\overline{\alpha}_{t-1}$; `// geometric averaging`
Output: $\overline{\alpha}_T$

of d terms, one per model parameter. One could subsample the parameters of the linear model in the data-fit term, but this would require us having to compute the full dataset's feature expansion at each step. This would be computationally intractable for the Jacobian basis functions we deal with in the later chapters of this thesis. Thus, for the weight-space formulation, we must fall-back on additive-noise and the reduced variance estimator corresponding to the objective in (4.17) is the best we can do.

Remark Can we apply the variance reduction strategy of Section 4.2.2 to the random coordinate estimator of the dual objective?
The variance reduction strategy presented earlier in this chapter amounts to moving the random noise in the sample-then-optimise targets from the data fit term to the regulariser. Here, we do this for the dual gradient (4.31). Letting the noisy targets be $z = f(X) + \mathcal{E}$, we move $\mathcal{E}$ to the regulariser while inverting its covariance by premultiplying by the noise precision b

$$\partial_\alpha L_d = (K\alpha - f(X) - \mathcal{E}) + b^{-1}\mathcal{E} = \underbrace{K\alpha - f(X)}_{\text{new fit gradient}} - \underbrace{b^{-1}(\alpha - b\mathcal{E})}_{\text{new reg. gradient}}.$$

Now, by inspection, it is clear that the random coordinate estimator, which subsamples the entries of the fit term and regulariser jointly, produces the same result when applied to both of the forms of the dual gradient written above. It is clear that we have nothing to gain by applying the variance reduction strategy.

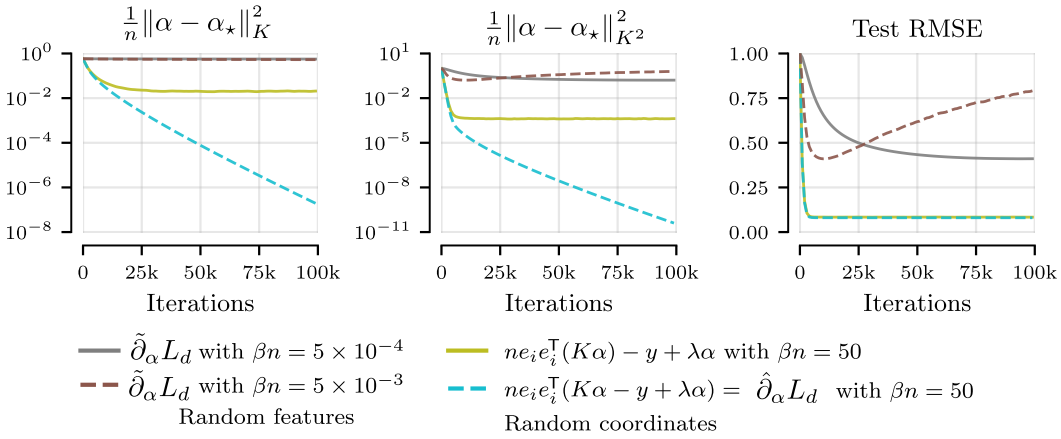


Fig. 4.4 A comparison of dual (stochastic) gradient descent on the POL data set with either random Fourier features or random coordinates, using batch size $r = 512$, momentum $\rho = 0.9$ and averaging parameter $\chi = 0.001$ (see Section 4.2.4 for explanation of latter two). Random features converge with $\beta n = 5 \times 10^{-4}$ but perform poorly, and diverge with a higher step-size. Random coordinates are stable with $\beta n = 50$ and show much stronger performance on all metrics. We include a version of random coordinates where only the $K\alpha$ term is subsampled: this breaks the multiplicative noise property, and results in an estimate which is worse on both the K-norm and the K^2-norm metric.

We term the combination of the dual gradient (4.31) with random coordinate estimation (4.33) as *Stochastic Dual Descent* (SDD). The corresponding algorithm is provided in algorithm 1. We discuss optimisation strategies in Section 4.2.4. We henceforth distinguish this algorithm from the one that uses the primal loss (4.4) and minibatching of only the fit term, as opposed to random coordinate estimation, by referring to the latter as SGD.

Demonstration: additive vs multiplicative noise

In Figure 4.4, we compare variants of stochastic dual descent with either random (Fourier) features or random coordinates. We see that random features, which produce high-variance additive noise, can only be used with very small step-sizes and have poor asymptotic performance. We test two versions of random coordinates: $\widehat{\partial}_\alpha L_d(\alpha)$, where, as presented, we subsample the whole gradient, and an alternative, $ne_i e_i^T(K\alpha) - y - b^{-1}\alpha$, where only the $K\alpha$ term is subsampled. While both are stable with much higher step-sizes than random features, the latter has worse asymptotic performance. This is a kind of *Rao-Blackwellisation trap:* introducing the known value of $-y + b^{-1}\alpha$ in place of its estimate $ne_i e_i^T(-y + b^{-1}\alpha)$ destroys the multiplicative property of the noise, making things worse, not better.

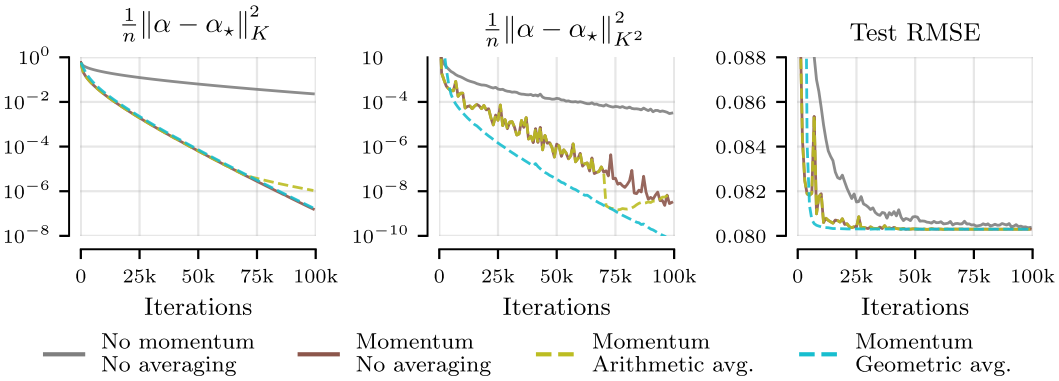


Fig. 4.5 Comparison of optimisation strategies for random coordinate estimator of the dual objective on the POL data set, using momentum $\rho = 0.9$, averaging parameter $\chi = 0.001$, batch size $r = 128$, and step-size $\beta n = 50$. Nesterov's momentum significantly improves convergence speed across all metrics. The dashed olive line, marked *arithmetic averaging*, shows the regular iterate up until 70k steps, at which point averaging commences and the averaged iterate is shown. Arithmetic iterate averaging slows down convergence in K-norm once enabled. Geometric iterate averaging, on the other hand, outperforms arithmetic averaging and unaveraged iterates throughout optimisation.

4.2.4 Getting the optimiser *right*

Momentum, or acceleration, is a range of modifications to the usual gradient descent updates that aim to improve the rate of convergence, in particular with respect to its dependence on the curvature of the optimisation problem (Polyak, 1964). We use Nesterov's momentum (Nesterov, 1983), as adapted for deep learning by Sutskever et al. (2013), because it is readily available in standard deep learning libraries. Algorithm 1 gives the precise updates. We use a momentum of $\rho = 0.9$ throughout. Comparing the plots in Figure 4.5, we see that momentum is vital on this problem, independently of iterate averaging.

When gradient updates are stochastic with additive noise and step-size is constant, the resulting iterates will bounce around the optimum rather than converging. In this setting, one can recover a convergent algorithm by arithmetically averaging the tail iterates, a procedure called *Polyak–Ruppert averaging* (Polyak, 1990; Polyak and Juditsky, 1992; Ruppert, 1988). While Polyak–Ruppert averaging is necessary with constant step-size and additive noise, it is not under multiplicative noise (Varre et al., 2021), and indeed can slow convergence. We recommend using *geometric averaging* instead, where we let $\overline{\alpha}_0 = \alpha_0$ and, at each step, compute

$$\overline{\alpha}_t = \chi\alpha_t + (1-\chi)\overline{\alpha}_{t-1} \quad \text{for an averaging parameter} \quad \chi \in (0, 1],$$

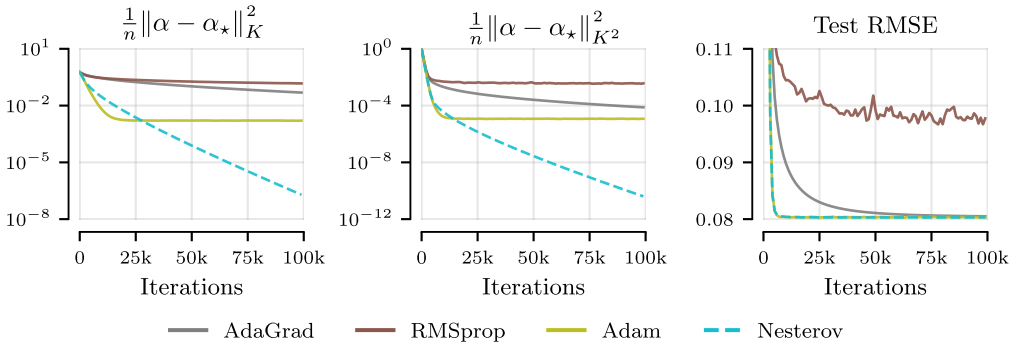


Fig. 4.6 Comparison of stochastic dual descent on POL with batch size $r = 512$ and averaging parameter $\chi = 0.001$, using different optimisers. While Adam and Nesterov perform similarly on Test RMSE, the latter has much closer convergence in both K-norm and K^2-norm

and return $\overline{\alpha}_T$. Geometric averaging is an anytime approach. It does not rely on fixed averaging-window size, and thus can be used in combination with early stopping, and the value of χ can be tuned adaptively. Here and throughout, we set $\chi = 100/T$, for T the total number of steps we perform. Figure 4.5 shows that geometric averaging outperforms both arithmetic averaging, and simply returning the last iterate α_T without any averaging.

Demonstration: Comparing Polyak momentum and geometric averaging with popular optimisers

In Figure 4.6, we report the performance of different optimisers on the dual problem. While algorithms such as AdaGrad, RMSprop, and Adam are designed to tackle problems with a non-constant curvature, the problem of sampling from a GP posterior is quadratic. Here, Nesterov-type momentum is theoretically rate-optimal. For all optimisers, we tune the step-size in a range of $[0.01, 100]$ and report the best performance; Adam with 0.05, AdaGrad with 10, RMSProp with 0.05, and Nesterov's momentum with 50. As predicted by the theory, Nesterov does best.

4.3 SGD for inference with inducing points

So far, our sampling objectives have presented linear cost in the dataset size. In the large-scale setting, algorithms with costs independent of the dataset size are often preferable. For GPs, this can be achieved through *inducing point posteriors* (Hensman et al., 2013; Titsias, 2009a), reviewed in Section 3.1.2, to which we now extend SGD sampling.

Let $Z = (z_1, z_2, \ldots, z_m) \in X^m$ be a set of $m \in \mathcal{N}$ inducing points.

By inspecting (3.12), we see that the optimal inducing point mean $\mu^{(Z)}_{f|Y}$ can be written

$$\mu^{(Z)}_{f|Y}(\cdot) = K_{(\cdot)Z}\alpha_\star = \sum_{j=1}^{m} \alpha_{*i} k(z_j, \cdot) \quad \alpha_\star = \arg\min_{\alpha \in \mathbb{R}^m} \sum_{i=1}^{n} [B]_{ii}(y_i - K_{x_i Z}\alpha)^2 + \|\alpha\|^2_{K_{ZZ}}, \tag{4.34}$$

and we can parameterise the uncertainty reduction term in the same way but with representer weights given by

$$\arg\min_{\alpha \in \mathbb{R}^m} \sum_{i=1}^{n} [B]_{ii}(f(x_i) + \varepsilon_i - K_{x_i Z}\alpha)^2 + \|\alpha\|^2_{K_{ZZ}} \tag{4.35}$$
$$\text{with} \quad f^{(Z)}(x_i) \sim \mathcal{N}(0, K_{x_i Z} K_{ZZ}^{-1} K_{Z x_i}) \quad \text{and} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}).$$

Derivation Derivation of inducing point sampling objectives (4.34) and (4.35).

Both expressions are derived in the same way, with only the targets we regress against changing between the objective for the variational posterior mean and samples. We part from the pathwise form of the 0-mean Kullback–Leibler-optimal inducing point GP

$$\begin{aligned}(f^{(Z)}|Y)(\cdot) &= \\ &= f(\cdot) + u^{(Z)}_{f|Y}(\cdot) - K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}(K_{XZ}K_{ZZ}^{-1}K_{ZX} + B^{-1})^{-1}(f^{(Z)}(X) + \mathcal{E}) \\ &\quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \qquad f \sim \text{GP}(0, k) \qquad f^{(Z)}(\cdot) = K_{(\cdot)Z}K_{ZZ}^{-1} f(Z).\end{aligned}$$

and apply the Woodbury identity to obtain

$$K_{(\cdot)Z}K_{ZZ}^{-1}K_{ZX}(K_{XZ}K_{ZZ}^{-1}K_{ZX} + B^{-1})^{-1}(f^{(Z)}(X) + \varepsilon) \tag{4.36}$$
$$= K_{(\cdot)Z}(K_{XZ}BK_{XZ} + K_{ZZ})^{-1}K_{ZX}B(f^{(Z)}(X) + \varepsilon) \tag{4.37}$$
$$= K_{(\cdot)Z}\alpha_\star. \tag{4.38}$$

Now, we recognize $(K_{XZ}BK_{XZ}+K_{ZZ})^{-1}K_{ZX}B(f^{(Z)}(X)+\varepsilon) = \alpha_\star$ as the expression for the optimiser of a ridge-regularised linear regression problem—see (2.7)—with parameters α, features K_{XZ} , Gaussian noise of covariance B^{-1}, and regulariser curvature K_{ZZ}. The targets are given by the random variable $(f^{(Z)}(X) + \mathcal{E})$.

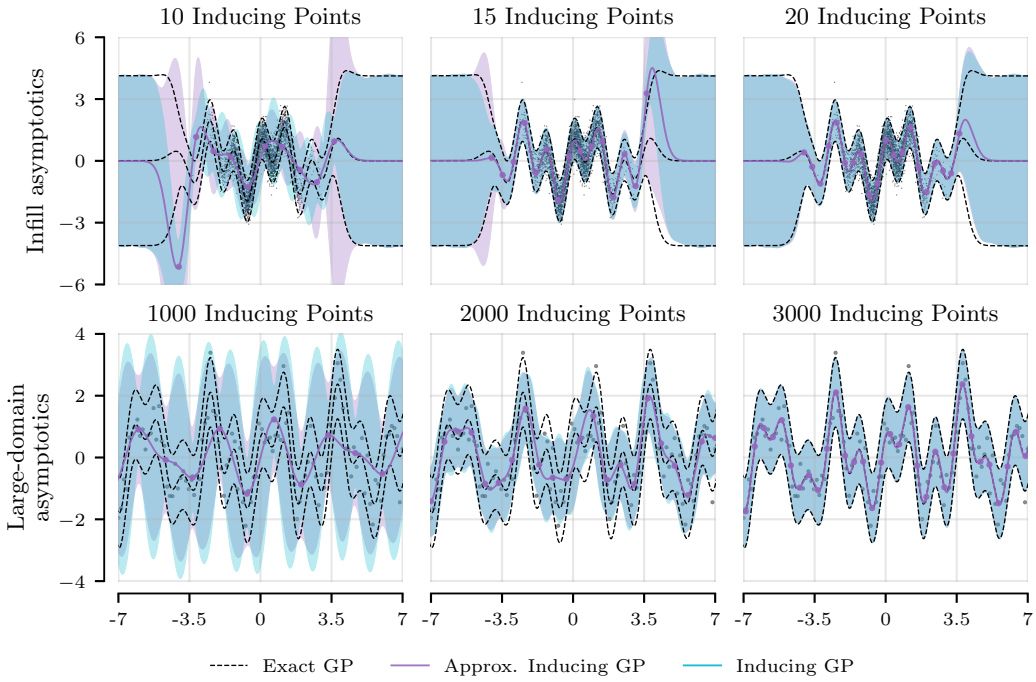


Fig. 4.7 Comparison of exact and approximate inducing point posteriors for a GP with squared exponential kernel and 10k data points generated using the true regression function $\sin(2x) + \cos(5x)$ under two different data-generation schemes: *infill asymptotics*, which considers $x_i \sim \mathcal{N}(0, 1)$, and *large-domain asymptotics*, which considers x_i on an evenly spaced grid with fixed spacing. We see that the approximation needed to apply inducing points is only inaccurate in situations where the inducing point posterior itself has significant error, which generally manifests itself as error bars that are larger than those of the exact posterior.

Exact implementation of (4.35) is precluded by the need to draw prior samples from a Gaussian with covariance $K_{XZ}K_{ZZ}^{-1}K_{ZX}$. This would require inverting K_{ZZ}, which presents cubic cost in m and may be poorly conditioned. However, we identify this matrix as a Nyström (i.e. low rank for $m < n$) approximation to K. Thus, we can approximate (4.35) by replacing $f^{(Z)} \sim \text{GP}(0, K_{(\cdot),Z}K_{Z,Z}^{-1}K_{Z,(\cdot)})$ with $f \sim \text{GP}(0, k(\cdot, \cdot))$, which can be, in turn, accurately approximated with random features (2.27). The error in approximating $f^{(Z)}$ with f is small when the number of inducing points m is large and the inducing points are close enough to the data. That is, whenever the inducing point GP is a good approximation to the posterior GP.

Remark on the Error in the Nyström Approximation $K_{X,Z}K_{Z,Z}^{-1}K_{Z,X} \approx K$
Figure 4.7 compares the KL-optimal inducing point posterior GP with that obtained

when taking the prior function samples which we fit with the representer weighed evaluation functionals to be $f(X)$ with $f \sim \mathrm{GP}(0, k)$ instead of $f^{(Z)}(X) = K_{XZ}K_{ZZ}^{-1}f(Z)$. This amounts to approximating the Nyström-type matrix $K_{X,Z}K_{Z,Z}^{-1}K_{Z,X}$ with its exact counterpart $K_{X,X}$. Both of these matrices become very similar if there is an inducing point placed sufficiently close to every data point. In practice, this tends to occur when an inducing point is placed within roughly a half-length-scale of every observation. This is effectively what is needed for inducing point methods to provide a good approximation of the exact GP. This is reflected in Figure 4.7, where we see that our approximate inducing point posterior differs from the exact inducing point posterior only in situations where the latter fails to be a good approximation to the exact GP in the first place. This manifests as the approximate method providing larger error bars. When the number of inducing points increases, both methods become indistinguishable from each other and the exact GP. Fortunately, the linear cost of SGD in the number of inducing points allows us to use a very large number of these in practice.

We now turn to stochastic estimation of the inducing point sampling objectives (4.34) and (4.35). Sadly, none of the tricks developed in this chapter are applicable. Since the curvature of the data-fit term $K_{ZX}BK_{XZ}$ differs from that of the regulariser K_{zz}, we can not apply stochastic dual descent. Additionally, the data fit term is a sum of n terms, while the regulariser is a sum of m terms; we can not apply the random coordinate estimator either. Finally, the low-variance estimator of Section 4.2.2, would require sampling the noise in the regularisation term from $\mathcal{N}(0, K_{zz}^{-1}K_{ZX}BK_{XZ}K_{zz}^{-1})$, which is also intractable for large numbers of inducing points. With this, we apply the simple minibatching plus random feature stochastic estimator given in (4.5) to the inducing point sampling objective.

The inducing point objectives differ from those presented in previous sections in that there are $\mathcal{O}(m)$ and not $\mathcal{O}(n)$ learnable parameters, and we may choose the value of m and locations Z freely. The cost of inducing point representer weight updates is thus $\mathcal{O}(sm)$, where s is the number of samples.

Demonstration: inducing point SGD

We demonstrate the inducing point variant of our method, on HOUSEELECTRIC, our largest dataset (n=2M). We select varying numbers of inducing points from the pool of train points. In particular, we use a K-nearest-neighbour algorithm to find and eliminate the points nearest to other points (in terms of euclidean distance). Figure 4.8 shows the time required for 100k SGD steps scales roughly linearly with inducing points. It takes 68m for full SGD and 50m,

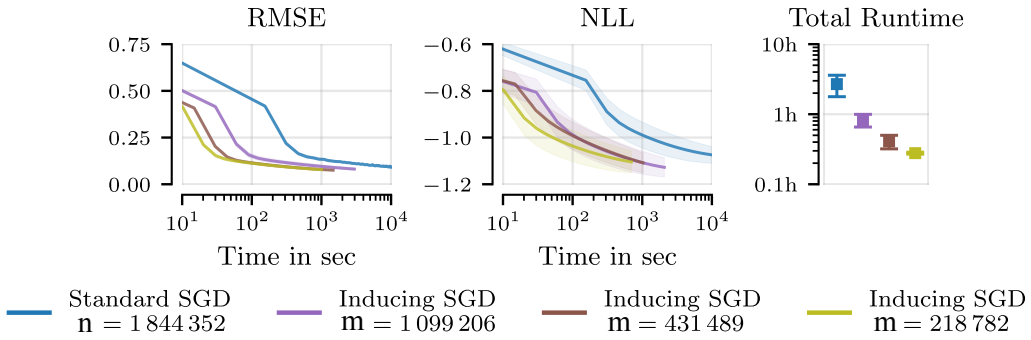


Fig. 4.8 Test RMSE and negative log-likelihood (NLL) obtained by SGD and its inducing point variants, for decreasing numbers of inducing points, given in the rightmost plot, as a function of time on an A100 GPU, on the HOUSEELECTRIC dataset ($n \approx 2$M).

25m, and 17m for m=1099m, 728k, and 218k, respectively. Performance in terms of RMSE and NLL degrades less than 10% even when using 218k points.

Model	Inducing SGD			Standard SGD	CG	SVGP
m	218 782	431 489	1 099 206	1 844 352	1 844 352	1 024
RMSE	**0.08 ± 0.00**	**0.08 ± 0.00**	**0.08 ± 0.01**	0.09 ± 0.00	0.87 ± 0.14	0.10 ± 0.02
Hours	0.28 ± 0.01	0.41 ± 0.09	0.83 ± 0.17	2.69 ± 0.91	2.62 ± 0.01	**0.04 ± 0.00**
NLL	**-1.10 ± 0.05**	**-1.11 ± 0.04**	**-1.13 ± 0.04**	**-1.09 ± 0.04**	2.07 ± 0.58	-0.94 ± 0.13

Table 4.1 Time to convergence (on an A100 GPU) and predictive performance for all approximate inference methods under consideration in this chapter, including inducing point SGD, on the HOUSEELECTRIC dataset. Experimental details are provided below, in Section 4.5.

Table 4.1 provides quantitative results for inducing point SGD on the HOUSEELECTRIC dataset. SGD's time to convergence is shown to scale roughly linearly in the number of inducing points. However, for this dataset, keeping only 10% of observations as inducing points and thus obtaining $10\times$ faster convergence leaves performance unaffected. This suggests the dataset can be summarised well by a small number of points. Indeed, SVGP obtains almost as strong performance as SGD in terms of RMSE with only 1024 inducing points. SVGP's NLL is weaker however, which is consistent with known issues of uncertainty overestimation when using a too small amount of inducing points. On the other hand, the large and potentially redundant nature of this dataset makes the corresponding optimisation problem ill-conditioned, hurting CG's performance.

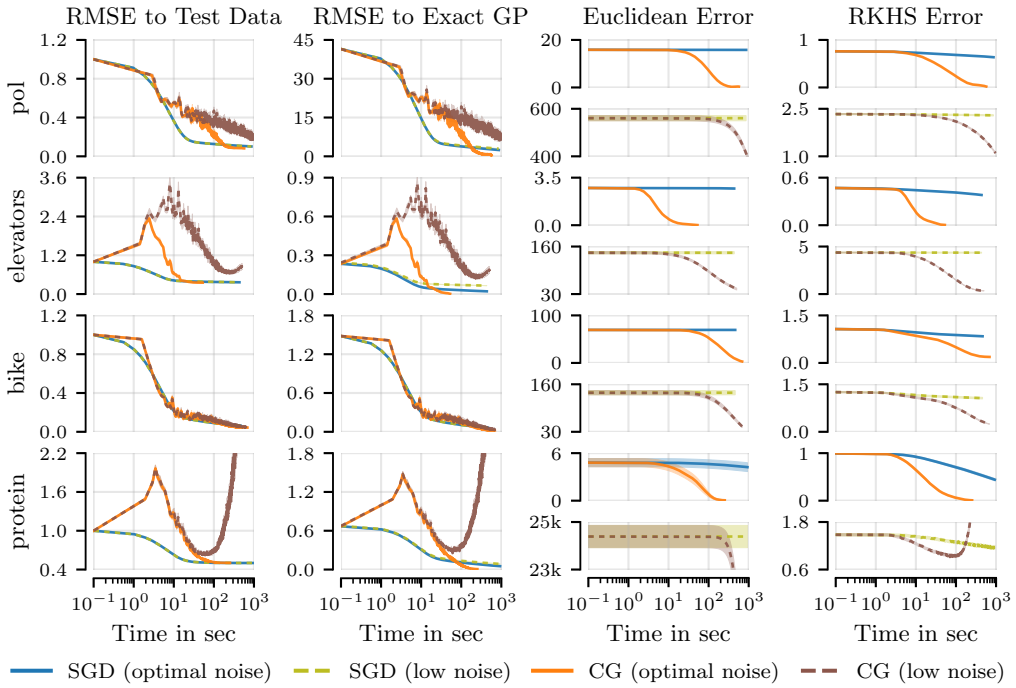


Fig. 4.9 Convergence of the GP posterior mean with SGD and CG as a function of time (on an A100 GPU) on the POL ($N \approx 15$k), ELEVATORS ($N \approx 16$k), BIKE ($N \approx 17$k) and PROTEIN ($N \approx 46$k) datasets, while setting the noise scale to (i) maximise exact GP marginal likelihood and (ii) to 10^{-3}, labelled *low noise*. We plot, in left-to-right order, test RMSE, RMSE to the exact GP mean at the test inputs, which is related to the K^2 norm $\|\alpha - \alpha_\star\|_{K^2}$, representer weight euclidean error $\|\alpha - \alpha^*\|$, and RKHS error $\|\mu_{f|Y} - \mu_{\text{SGD}}\|_{\mathcal{H}} = \|\alpha - \alpha_\star\|_K$, i.e. K norm. In the latter two plots, the low-noise setting is shown on the bottom.

4.4 Analysing the implicit bias of stochastic gradient descent

We have detailed an SGD-based scheme for obtaining approximate samples from a posterior Gaussian process. Despite SGD's significantly lower cost per-iteration than CG, its convergence to the true optima, shown in Figure 4.9, is much slower in both Euclidean representer weight space, and the reproducing kernel Hilbert space (RKHS) induced by the kernel. Despite this, the predictions obtained by SGD are very close to those of the exact GP, and effectively achieve the same test RMSE. Moreover, Figure 4.10 shows the SGD posterior on a 1D toy task exhibits error bars of the correct width close to the data, and which revert smoothly to the prior far away from the data. Empirically, differences between the SGD and exact posteriors concentrate at the borders of data-dense regions.

We now argue the behavior seen in Figure 4.10 is a general feature of SGD: one can expect it to obtain good performance even in situations where it does not converge to the exact solution.

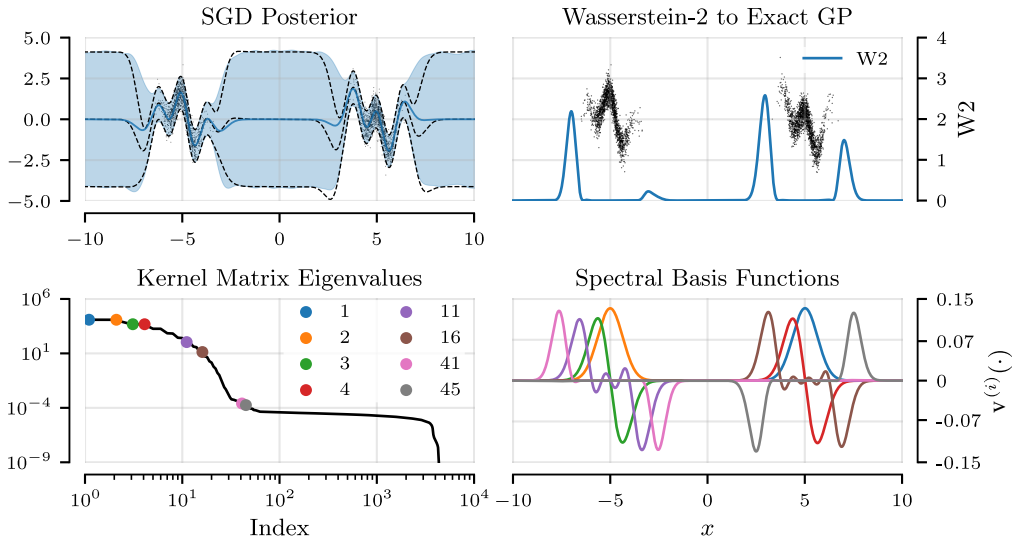


Fig. 4.10 SGD error and spectral basis functions. Top-left: SGD (blue) and exact GP (black, dashed) fit to a n=10k, toy regression dataset. Top-right: 2-Wasserstein distance (W2) between both processes' marginals. The W2 values are low near the data (interpolation region) and far away from the training data. The error concentrates at the edges of the data (extrapolation region). Bottom: The low-index spectral basis functions lie on the interpolation region, where the W2 error is low, while functions of index 10 and larger lie on the extrapolation region where the error is large.

Consider posterior function samples in pathwise form, namely $(f|Y)(\cdot) = f(\cdot) + K_{(\cdot)X}\alpha$, where $f \sim \mathrm{GP}(0, k)$ is a prior function sample and α are the learnable representer weights. We characterise the behavior of SGD-computed approximate posteriors by splitting the input space $\mathcal{X}$ into 3 regions, which we call the *far-away*, *interpolation*, and *extrapolation* regions. This is done as follows.

(I) The Far-away Region. This corresponds to points sufficiently distant from the observed data. Here, for kernels that decay over space, the evaluation functionals $k(x_i, \cdot)$ go to zero. Thus, both the true posterior and any approximations formulated pathwise revert to the prior. More precisely, let $X = \mathbb{R}^d$, let k satisfy $\lim_{c\to\infty} k(x', c \cdot x) = 0$ for all x' and x in $\mathcal{X}$, and let $(f|Y)(\cdot)$ be given by $(f|Y)(\cdot) = f(\cdot) + K_{(\cdot)X}\alpha$, with $\alpha \in \mathbb{R}^n$. Then, for any fixed α, any choice of $x \in \mathcal{X}$, it follows immediately that $\lim_{c\to\infty}(f|Y)(c \cdot x) = f(c \cdot x)$. Therefore, SGD cannot incur error in regions which are sufficiently far away from the data. This effect is depicted in Figure 4.10.

(II) The Interpolation Region. This includes points close to the training data. We characterise this region through subspaces of the RKHS, where we show SGD incurs small error.

Let $K = V\Lambda V^T$ be the eigendecomposition of the kernel matrix. We index the eigenvalues $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ in descending order. Define the *spectral basis functions* as eigenvector-weighed linear combinations of evaluation functionals

$$v^{(i)}(\cdot) = \sum_{j=1}^{n} \frac{[V]_{ji}}{\sqrt{\lambda_i}} k(x_j, \cdot). \tag{4.39}$$

These functions are orthonormal with respect to the RKHS inner product. To characterise them further, consider the following characterisation of eigenvalues and eigenvectors in the RKHS $\mathcal{H}$

$$v^{(i)}(\cdot) = \arg\max_{v \in \mathcal{H}} \left\{ \sum_{i=1}^{n} v(x_i)^2 : \|v\|_{\mathcal{H}} = 1, \langle v, v^{(j)} \rangle = 0, \forall j < i \right\}. \tag{4.40}$$

This tells us that the top spectral basis function, $v^{(1)}(\cdot)$, is a function of fixed RKHS norm—that is, of fixed degree of smoothness, as defined by the kernel k—which takes maximal values at the observations $x_1, .., x_n$. Thus, $v^{(1)}$ will be large near clusters of observations. The same will be true for the subsequent spectral basis functions, which also take maximal values at the observations, but are constrained to be RKHS-orthogonal to previous spectral basis functions. Figure 4.10 confirms that the top spectral basis functions are indeed centred on the observed data.

Empirically, SGD matches the true posterior in the region of the top spectral basis functions, i.e. in the data dense regions. We now formalise this observation by showing that SGD converges quickly in the directions spanned by spectral basis functions with large eigenvalues. For this, we consider the primal objective (4.2) with minibatching for the data fit but no random feature estimation of the regulariser; this provides us with a sub-Gaussian additive noise estimator of the gradient. To simplify the analysis, we assume the use of arithmetic iterate averaging, as opposed to geometric averaging, and no momentum. Let $\text{proj}_{v^{(i)}}(\cdot)$ be the orthogonal projection onto the subspace spanned by $v^{(i)}$.

Proposition 1. *Let $\delta > 0$. Let $B^{-1} = b^{-1}I$ for $b^{-1} > 0$. Let μ_{SGD} be the predictive mean function obtained by arithmetically-averaged SGD after t steps, starting from an initial set of representer weights equal to zero, and using a sufficiently small learning rate of $0 < \beta < \frac{b^{-1}}{\lambda_1(\lambda_1 + b^{-1})}$. Assume the stochastic estimate of the gradient is G-sub-Gaussian. Then, with probability $1 - \delta$, we have for $i = 1, .., N$ that*

$$\left\| \textit{proj}_{v^{(i)}} \mu_{f|Y} - \textit{proj}_{v^{(i)}} \mu_{SGD} \right\|_{\mathcal{H}} \leq \frac{1}{\sqrt{\lambda_i t}} \left(\frac{b\|Y\|_2}{\eta} + G\sqrt{\frac{2}{t} \log \frac{N}{\delta}} \right). \tag{4.41}$$

This is an extension of a standard result on the convergence of SGD (LeCun et al., 1992) to the span of the spectral basis functions. For the proof, as well as an additional pointwise convergence bound, and a variant that handles projections onto general subspaces spanned by basis functions, we refer to Appendix E of Lin et al. (2023b). In general, we expect G to be at most $\mathcal{O}(\lambda_1^2 \|Y\|_\infty)$ with high probability. An analogous result is straightforward to obtain for the dual gradient (4.31). It allows us to raise our learning rate to $0 < \beta < \frac{1}{\lambda_1 + b^{-1}}$.

The result extends immediately from the posterior mean to posterior samples. As consequence, *SGD converges to the posterior GP quickly in the data-dense region*, namely where the spectral basis functions corresponding to large eigenvalues are located. Since convergence speed on the span of each basis function is independent of the magnitude of the other basis functions' eigenvalues, SGD can perform well even when the kernel matrix is ill-conditioned. This is shown in Figure 4.9.

(III) The Extrapolation Region. This can be found by elimination from the input space of the far-away and interpolation regions, in both of which SGD incurs low error. Consider the spectral basis functions $v^{(i)}(\cdot)$ with small eigenvalues. By orthogonality of $v^{(1)}, .., v^{(N)}$, such functions cannot be large near the observations while retaining a prescribed norm. Their mass is therefore placed away from the observations. SGD converges slowly in this region, resulting in a large error in its solution in both a Euclidean and RKHS sense, as seen in Figure 4.9. Fortunately, due to the lack of data in the extrapolation region, the excess test error incurred due to SGD nonconvergence may be low, resulting in benign nonconvergence (Zou et al., 2021). Figure 4.10 shows the Wasserstein distance to the exact GP predictions is high in this region, as SGD tends to return small representer weights, thereby reverting to the prior.

Demonstration: resilience to ill conditioning

We explore how this section's result affects the algorithms under consideration by setting a small isotropic noise variance of $b^{-1} = 10^{-6}$ and running them on our set of UCI regression datasets. Table 4.2 shows the performance of CG severely degrades on all datasets. SVGP diverges for all datasets. SGD's results remain essentially-unchanged. This is because the noise only changes the smallest kernel matrix eigenvalues substantially and these do not affect convergence in the direction of the top spectral basis functions. This mirrors results presented in Figure 4.9.

	Dataset	POL	ELEVATORS	BIKE	PROTEIN	KEGGDIR	3DROAD	SONG	BUZZ	HOUSEELEC
	N	15000	16599	17379	45730	48827	434874	515345	583250	2049280
RMSE	SGD	0.13 ± 0.00	0.38 ± 0.00	0.11 ± 0.00	$\mathbf{0.51 \pm 0.00}$	0.12 ± 0.00	$\mathbf{0.11 \pm 0.00}$	$\mathbf{0.80 \pm 0.00}$	0.42 ± 0.01	$\mathbf{0.09 \pm 0.00}$
	CG	$\mathbf{0.08 \pm 0.00}$	$\mathbf{0.35 \pm 0.00}$	$\mathbf{0.04 \pm 0.00}$	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.08 \pm 0.00}$	0.15 ± 0.01	0.85 ± 0.03	1.41 ± 0.08	0.87 ± 0.14
	SVGP	0.10 ± 0.00	0.37 ± 0.00	0.08 ± 0.00	0.62 ± 0.00	0.10 ± 0.00	0.64 ± 0.01	0.82 ± 0.00	$\mathbf{0.34 \pm 0.00}$	0.10 ± 0.02
RMSE†	SGD	$\mathbf{0.13 \pm 0.00}$	$\mathbf{0.38 \pm 0.00}$	0.11 ± 0.00	$\mathbf{0.51 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.11 \pm 0.00}$	$\mathbf{0.80 \pm 0.00}$	$\mathbf{0.42 \pm 0.01}$	$\mathbf{0.09 \pm 0.00}$
	CG	0.16 ± 0.01	0.68 ± 0.09	$\mathbf{0.05 \pm 0.01}$	3.03 ± 0.23	9.79 ± 1.06	0.34 ± 0.02	0.83 ± 0.02	5.66 ± 1.14	0.93 ± 0.19
	SVGP	—	—	—	—	—	—	—	—	—

Table 4.2 Mean and std. err. of the test RMSE and low-noise test RMSE (†) obtained by the GP predictive mean computed with SGD, CG and SVGP. The latter method is omitted for the low noise setting, where it fails to run. Metrics are reported for the datasets normalised to zero mean and unit variance. The full experimental setup is described below in Section 4.5.

4.5 Experiments and benchmarks

We now turn to empirical evaluation of SGD GPs and SDD GPs. We compare these with the two most popular scalable Gaussian process techniques: preconditioned conjugate gradient (CG) optimisation (Gardner et al., 2018; Wang et al., 2019) and sparse stochastic variational inference (SVGP) (Hensman et al., 2013; Titsias, 2009a). We employ the JAX.SCIPY CG implementation and follow Wang et al. (2019) in using a pivoted Cholesky preconditioner of size 100. Our preconditioner implementation resembles the implementation of the TENSORFLOW PROBABILITY library. For a small subset of datasets, we find the preconditioner to lead to slower convergence, and we report the results for conjugate gradients without preconditioning instead. We employ the GPJax (Pinder and Dodd, 2022) SVGP implementation and initialise inducing point locations with the K-means algorithm. In all SGD and SDD experiments, we use a Nesterov momentum value of $\rho = 0.9$ and geometric averaging with $\chi = 100/T$ for T the total number of steps. The latter is chosen on a task-dependent basis. For SGD, at each step, we draw 100 random features to unbiasedly estimate the regulariser term. When drawing posterior samples with all methods we use pathwise conditioning with 2000 random Fourier features to draw each prior function.

4.5.1 UCI benchmark datasets

We first compare SGD-based predictions with baselines in terms of predictive performance, scaling of computational cost with problem size, and robustness to the ill-conditioning of linear systems. Following Wang et al. (2019), we consider 9 datasets from the UCI repository (Dua and Graff, 2017) ranging in size from $n = 15$k to $n \approx 2$M datapoints and input

dimensionality from $d' = 3$ to $d' = 90$. We report mean and standard deviation over five 90%-train 10%-test splits for the small and medium datasets, and three splits for the largest dataset.

GP hyperparameters We use a zero prior mean function and the Matérn-$3/2$ kernel, and share hyperparameters across all methods, including baselines. For each dataset, we choose a homoscedastic Gaussian noise variance, a single kernel variance, and a separate length scale per input dimension. For datasets with less than 50k observations, we tune these hyperparameters to maximise the exact GP marginal likelihood (2.43). The cubic cost of this procedure makes it intractable at a larger scale: instead, for datasets with more than 50k observations, we obtain hyperparameters using the following procedure:

1. From the training data, select a *centroid* data point uniformly at random.
2. Select the subset of 10k data points with the smallest Euclidean distance to the centroid.
3. Find hyperparameters by maximizing the exact GP marginal likelihood using this subset of data.
4. Repeat the preceding steps for 10 different centroids, and average the resulting hyperparameters.

This approach avoids aliasing bias Barbano et al. (2022b) due to data subsampling and is tractable for large datasets.

Inference method hyperparameters We run SGD for 100k steps, with a fixed batch size of 512 for both the mean function and samples. For all regression experiments, we use a learning rate of 0.5 to estimate the mean function representer weights, and a learning rate of 0.1 to draw samples. For SDD, we use step-sizes $100\times$ larger than SGD, except for ELEVATORS, KEGGDIRECTED and BUZZ, where this causes divergence and we use $10\times$ larger step-sizes instead. We run CG to a tolerance of 0.01, except for the 4 largest data sets, where we stop CG after 100 iterations—this still provides CG with a larger compute budget than first-order methods. For SVGP, we use $3,000$ inducing points for the smaller five data sets and $9,000$ for the larger four, so as to match the runtime of the other methods. For all methods, we estimate predictive variances for log-likelihood computations from 64 function samples drawn using pathwise conditioning.

	Data	POL	ELEVATORS	BIKE	PROTEIN	KEGGDIR	3DROAD	SONG	BUZZ	HOUSEELEC
	Size	15k	17k	17k	46k	49k	435k	515k	583k	2M
RMSE	SDD	**0.08 ± 0.00**	**0.35 ± 0.00**	**0.04 ± 0.00**	**0.50 ± 0.01**	**0.08 ± 0.00**	**0.04 ± 0.00**	**0.75 ± 0.00**	**0.28 ± 0.00**	**0.04 ± 0.00**
	SGD	0.13 ± 0.00	0.38 ± 0.00	0.11 ± 0.00	0.51 ± 0.00	0.12 ± 0.00	0.11 ± 0.00	0.80 ± 0.00	0.42 ± 0.01	0.09 ± 0.00
	CG	**0.08 ± 0.00**	**0.35 ± 0.00**	**0.04 ± 0.00**	**0.50 ± 0.00**	**0.08 ± 0.00**	0.18 ± 0.02	0.87 ± 0.05	1.88 ± 0.19	0.87 ± 0.14
	SVGP	0.10 ± 0.00	0.37 ± 0.00	0.08 ± 0.00	0.57 ± 0.00	0.10 ± 0.00	0.47 ± 0.01	0.80 ± 0.00	0.32 ± 0.00	0.12 ± 0.00
Time (min)	SDD	1.88 ± 0.01	1.13 ± 0.02	1.15 ± 0.02	1.36 ± 0.01	1.70 ± 0.00	**3.32 ± 0.01**	**185 ± 0.56**	**207 ± 0.10**	**47.8 ± 0.02**
	SGD	2.80 ± 0.01	2.07 ± 0.03	2.12 ± 0.04	2.87 ± 0.01	3.30 ± 0.12	6.68 ± 0.02	190 ± 0.61	212 ± 0.15	69.5 ± 0.06
	CG	**0.17 ± 0.00**	**0.04 ± 0.00**	**0.11 ± 0.01**	**0.16 ± 0.01**	**0.17 ± 0.00**	13.4 ± 0.01	192 ± 0.77	244 ± 0.04	157 ± 0.01
	SVGP	11.5 ± 0.01	11.3 ± 0.06	11.1 ± 0.02	11.1 ± 0.02	11.5 ± 0.04	152 ± 0.15	213 ± 0.13	209 ± 0.37	154 ± 0.12
NLL	SDD	**-1.18 ± 0.01**	**0.38 ± 0.01**	-2.49 ± 0.09	**0.63 ± 0.02**	**-0.92 ± 0.11**	**-1.70 ± 0.01**	**1.13 ± 0.01**	**0.17 ± 0.06**	**-1.46 ± 0.10**
	SGD	-0.70 ± 0.02	0.47 ± 0.00	-0.48 ± 0.08	0.64 ± 0.01	-0.62 ± 0.07	-0.60 ± 0.00	1.21 ± 0.00	0.83 ± 0.07	-1.09 ± 0.04
	CG	**-1.17 ± 0.01**	**0.38 ± 0.00**	**-2.62 ± 0.06**	**0.62 ± 0.01**	**-0.92 ± 0.10**	16.3 ± 0.45	1.36 ± 0.07	2.38 ± 0.08	2.07 ± 0.58
	SVGP	-0.67 ± 0.01	0.43 ± 0.00	-1.21 ± 0.01	0.85 ± 0.01	-0.54 ± 0.02	0.60 ± 0.00	1.21 ± 0.00	0.22 ± 0.03	-0.61 ± 0.01

Table 4.3 Root mean square error (RMSE), compute time (on an A100 GPU), and negative log-likelihood (NLL), for 9 UCI regression tasks for all methods considered. We report mean values and standard error across five 90%-train 10%-test splits for all data sets, except the largest, where three splits are used. Targets are normalised to zero mean and unit variance. This work denoted by SDD.

Results The results, reported in Table 4.3, show that SDD matches or outperforms all baselines on all UCI data sets in terms of root mean square error of the mean prediction across test data. SDD strictly outperforms SGD on all data sets and metrics, matches CG on the five smaller data sets, where the latter reaches tolerance, and outperforms CG on the four larger data sets. The same holds for the negative log-likelihood metric (NLL), except on BIKE, where CG marginally outperforms SDD. Since SDD requires only one matrix-vector multiplication per step, as opposed to two for SGD, it provides about 30% wall-clock time speed-up relative to SGD. Although we run SDD for 100k iterations to match the SGD baseline, SDD often converges earlier than that.

Remark Why SGD and SDD outperform CG on large problems
SGD and SDD present two key advantages which make them perform well on very large scale tasks. The first is their relative insensitivity to problem conditioning (see Section 4.4). We only expect SGD to converge in the direction of the top eigenvectors of the curvature matrix. Poor conditioning will make it converge slowly in the bottom eigendirections, but optimisation noise would prevent convergence in those directions anyway. On the other hand, CG's runtime is heavily determined by conditioning (see Section 3.2.2). The second is SGD and SDD's compatibility with early stopping. From Figure 4.9, we see that SGD makes the vast majority of its progress in prediction space

in its first few iterations, improving roughly monotonically with the number of steps. Thus, early stopping after 100k iterations incurs only moderate errors. In contrast, CG monotonically decreases euclidean error and error measured in the RKHS norm but its initial steps actually increase test error (which is more related to the K^2 norm), resulting in very poor performance if stopped too early.

4.5.2 Large-scale Bayesian optimisation

A fundamental goal of scalable Gaussian processes is to produce uncertainty estimates useful for sequential decision making. Motivated by problems in large-scale recommender systems, where both the initial dataset and the total number of users queried are simultaneously large (Elahi et al., 2016; Rubens et al., 2015), we benchmark SGD on a large-scale Bayesian optimisation task. We draw a target function from a GP prior $f \sim \mathrm{GP}(0, k)$ and optimise it on $\mathcal{X} = [0, 1]^{d'}$ using parallel Thompson sampling (Hernández-Lobato et al., 2017), which we described in Section 2.3.3. We use an acquisition batch size of 1000 samples, and maximise them with a multi-start gradient descent-based approach described in Section 2.3.3[3]. We run 30 acquisition steps, acquiring a total of $30k$ observations. We set the search space dimensionality to $d' = 8$, the largest considered by Wilson et al. (2020), and initialise all methods with the same dataset of 50k observations sampled uniformly at random from $\mathcal{X}$. To eliminate model misspecification confounding, we use a Matérn-$3/2$ kernel and consider length scales of $(0.1, 0.2, 0.3, 0.4, 0.5)$ for both the target function and our models. For each length scale, we repeat the experiment for 10 seeds.

In large-scale Bayesian optimisation, training and posterior function optimisation costs can become significant, and predictions may be needed on demand. For this reason, we include two variants of the experiment, one with a small compute budget, where SGD and SDD are run for 15k steps, SVGP is given 20k steps and CG is run for 10 steps, and one with a large budget, where all methods are run for 5 times as many steps. We present the results on this task, broken down by lengthscale value, in Figure 4.11. In both large and small compute settings, and across lenghtscales, SDD makes the most progress, in terms of maximum value found, while using the least compute. Unlike SVGP and CG, The performance of SDD and SGD degrades gracefully when compute budget is limited. Here, SVGP performs well—on par with SGD—in the large length scale setting, where many observations can likely be summarised with 1024 inducing points. CG suffers from slow convergence due to ill-conditioning here. On the other hand, CG performs on par with SGD

[3]Please refer to Appendix A.3 of Lin et al. (2023b) for more details on our function maximisation strategy.

in the better-conditioned small length scale setting, while SVGP suffers. In the large compute setting, all methods perform similarly per acquisition step for all length scales except the small one, where SVGP suffers.

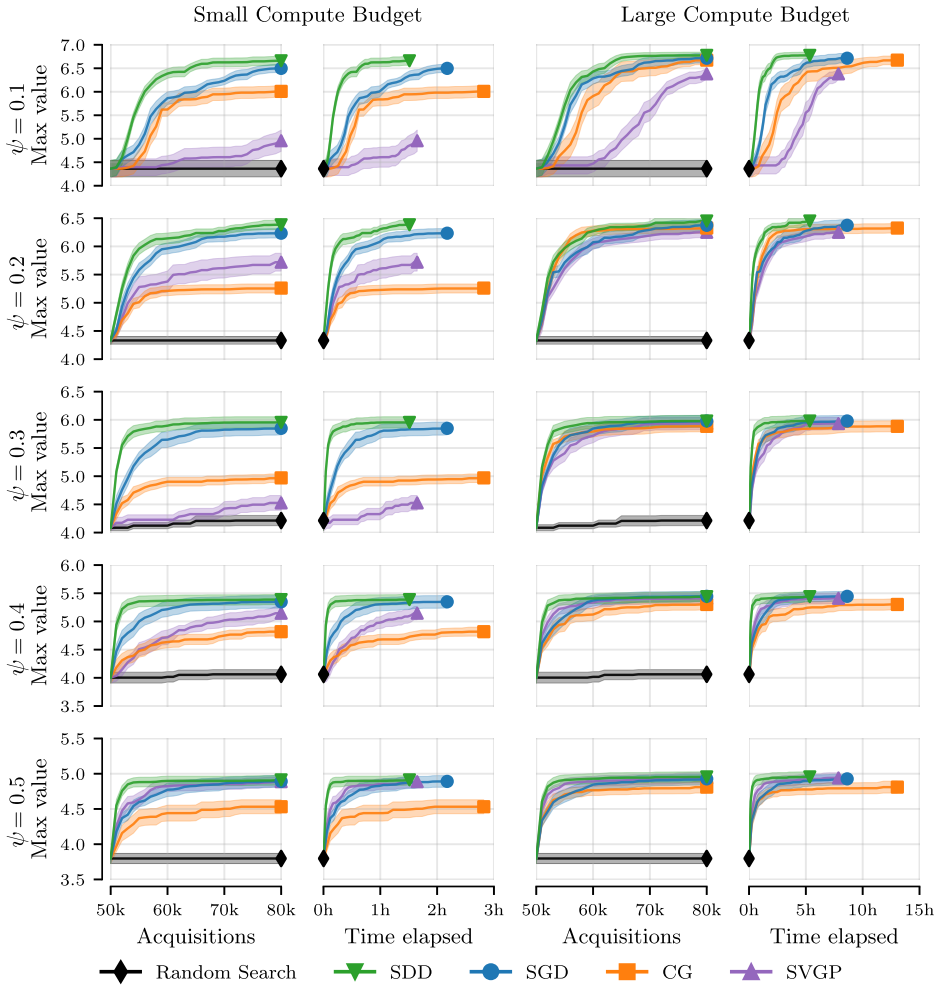


Fig. 4.11 Maximum function values, with mean and standard error across 10 seeds, obtained by parallel Thompson sampling, for functions with different length-scales ψ, plotted as functions of acquisition steps and the compute time on an A100 GPU. All methods share an initial data set of 50k points, and take 30 Thompson steps, acquiring a batch of 1000 points in each. The algorithms perform differently across the length-scales: CG performs better in settings with smaller length-scales, which give better conditioning; SVGP tends to perform better in settings with larger length-scales and thus higher smoothness; SGD and SDD perform well in both settings.

4.5.3 Molecule-protein binding affinity prediction

The binding affinity between a molecule and certain proteins is a widely used preliminary filter in drug discovery (Pinzi and Rastelli, 2019), and machine learning is increasingly used to estimate this quantity (Yang et al., 2021). In this final experiment, we show that Gaussian processes with SDD are competitive with graph neural networks for binding affinity prediction.

Dataset setup We use the DOCKSTRING regression benchmark of García-Ortegón et al. (2022), which contains five tasks, corresponding to five different proteins. The inputs are the graph structures of 250k candidate molecules, and the targets are real-valued affinity scores from the docking simulator *AutoDock Vina* (Trott and Olson, 2010). We perform all of the preprocessing steps for this benchmark outlined by García-Ortegón et al. (2022), including limiting the maximum docking score to 5. For each protein, we use a standard train-test splits of 210k and 40k molecules, respectively. These were produced by structure-based clustering to avoid similar molecules from occurring both in the train and test set. We use Morgan fingerprints of dimension 1024 (Rogers and Hahn, 2010) to represent the molecules.

Primer on fingerprints, Tanimoto Kernel and its random features Molecular fingerprints are a way to encode the structure of molecules by indexing sets of subgraphs present in a molecule. There are many types of fingerprints. Morgan fingerprints represent the subgraphs up to a certain radius around each atom in a molecule (Rogers and Hahn, 2010). The fingerprint can be interpreted as a sparse vector of counts, analogous to a 'bag of words' representation of a document. Accordingly, the Tanimoto coefficient $\mathfrak{T}(x, x')$, also called the Jaccard index, is a way to measure similarity between fingerprints, given by

$$\mathfrak{T}(x, x') = \frac{\sum_i \min(x_i, x'_i)}{\sum_i \max(x_i, x'_i)}.$$

This function is a valid kernel and has a known random feature expansion using random hashes (Tripp et al., 2023). We use this kernel for our GPs. The feature expansion builds upon prior work for fast retrieval of documents using random hashes that approximate the Tanimoto coefficient; that is, a distribution $P_\mathfrak{h}$ over hash functions $\mathfrak{h}$ such that

$$P_(\mathfrak{h}(\mathfrak{h}(x) = \mathfrak{h}(x')) = \mathfrak{T}(x, x') \ .$$

Per Tripp et al. (2023), we extend such hashes into random *features* by using them to index a random tensor whose entries are independent Rademacher random variables. We use the random hash of Ioffe (2010).

Gaussian process Setup As the Tanimoto kernel itself has no hyperparameters, the only kernel hyperparameters are a constant scaling factor $a^{-1} > 0$ for the kernel, the noise variance b^{-1}, and a constant GP prior mean μ_0 (the Gaussian process regresses on $y - \mu_0$ in place of y). These were chosen by Tripp et al. (2023) by maximising the evidence of an exact GP given a randomly chosen subset of the data and held constant during the optimisation of the inducing points. The values are given in Table 4.4. The same values are also used for SGD and SDD to ensure that the differences in accuracy are solely due to the quality of the GP posterior approximation. The SGD method uses 100-dimensional random features for the regulariser.

Data	ESR2	F2	KIT	PARP1	PGR
a^{-1}	0.497	0.385	0.679	0.560	0.630
b^{-1}	0.373	0.049	0.112	0.024	0.332
μ_0	-6.79	-6.33	-6.39	-6.95	-7.08

Table 4.4 Hyperparameters for all Gaussian process methods used in the molecule-protein binding affinity experiments of Section 4.5.3.

Results

In Table 4.5, following García-Ortegón et al. (2022), we report R^2 values. Alongside results for SDD and SGD, we incldue results from García-Ortegón et al. (2022) for XGBoost, and for two graph neural networks, MPNN (Gilmer et al., 2017) and Attentive FP (Xiong et al., 2019), the latter of which is the state-of-the-art for this task. We also include the results for SVGP reported by Tripp et al. (2023). These results show that SDD matches the performance of Attentive FP on the ESR2 and FP2 proteins, and comes close on the others. To the best of our knowledge, this is the first time Gaussian processes have been shown to be competitive on a large-scale molecular prediction task.

Method	ESR2	F2	KIT	PARP1	PGR
Attentive FP†	**0.627**	**0.880**	**0.806**	**0.910**	**0.678**
MPNN†	0.506	0.798	0.755	0.815	0.324
XGBoost†	0.497	0.688	0.674	0.723	0.345

Method	ESR2	F2	KIT	PARP1	PGR
SDD	**0.627**	**0.880**	0.790	0.907	0.626
SGD	0.526	0.832	0.697	0.857	0.408
SVGP‡	0.533	0.839	0.696	0.872	0.477

Table 4.5 Test set R^2 scores obtained for each target protein on the DOCKSTRING molecular binding affinity prediction task. Results with $(\cdot)^\dagger$ are from García-Ortegón et al. (2022), those with $(\cdot)^\ddagger$ are from Tripp et al. (2023). SVGP uses 1000 inducing points. SDD denotes this work.

4.6 Discussion

In this chapter, we explored using stochastic gradient algorithms to approximately compute Gaussian process posterior means and function samples at scale. We derived optimisation objectives with linear and sublinear (via inducing points) cost for both. We studied variance reduction techniques for the posterior sampling objective and also its conditioning. The latter investigation led to the development of stochastic dual descent, a specialised first-order stochastic optimisation algorithm for Gaussian processes. To design this algorithm, we combined a number of ideas from the optimisation literature with Gaussian-process-specific ablations, arriving at an algorithm which is simultaneously simple and matches or exceeds the performance of relevant baselines. We showed that SGD can produce accurate predictions, even in cases when it is early stopped and does not converge to an optimum. We developed a spectral characterisation of the effects of non-convergence, showing that it manifests itself mainly through error in an extrapolation region located away—but not too far away—from the observations. We benchmarked SGD and SDD, showing they yield strong performance on standard regression benchmarks and on a large-scale Bayesian optimisation benchmark. SDD matches the performance of state-of-the-art graph neural networks on a molecular binding affinity prediction task.

Being able to perform posterior inference in large-scale linear models, through stochastic optimisation, is the first step towards tackling the ultimate goal of this thesis: performing Bayesian reasoning with large-scale neural networks. Chapter 5 will further pursue this goal by studying connections between linear models and neural networks through the linearised Laplace approximation. In particular, the next chapter will focus on hyperparameter optimisation using the model evidence. Building upon this, Chapter 6 will use the methods introduced in this chapter to scale Bayesian inference to large-scale linearised NNs. For this, we will have to work with the non-kernelised setting, where SDD is not-applicable. We will rely on SGD instead.

Chapter 5

Adapting the linearised Laplace model evidence for modern deep learning

Model selection and uncertainty estimation are two important open problems in deep learning. The former aims to select network hyperparameters and architectures without costly cross-validation (Immer et al., 2021a, 2022; Mackay, 1992a). The latter provides a measure of fidelity of network predictions that can be used in downstream tasks such as experimental design (Barbano et al., 2022b), sequential decision making (Janz et al., 2019), and in safety-critical settings (Fridman et al., 2019). This thesis does not attempt to compute exact Bayesian posterior credible regions or the exact model evidence for NNs. This is likely impossible when dealing with large-scale networks. Instead, we will sacrifice orthodoxy and pursue Bayesian-inspired methods that scale well and provide good results. To this end, we focus on a classical approximate approach to these two problems: the linearised Laplace method (Mackay, 1992a), which has recently been shown to be one of the best performing methods for approximate inference in neural networks (Daxberger et al., 2021a,b; Immer et al., 2021b; Khan et al., 2019a; Kristiadi et al., 2020).

Linearised Laplace approximates the output of a neural network (NN) with a first order Taylor expansion (a linearisation) around optimal NN parameters. It then uses standard linear-model-type error bars to approximate the uncertainty in the output of the NN, while retaining the NN point-estimate as the predictive mean. The latter feature means that, unlike other Bayesian deep learning procedures, the linearised Laplace uncertainty estimates do not come at the cost of the accuracy of the predictive mean (Antorán et al., 2020; Ashukha et al., 2020; Snoek et al., 2019a). A downside of the method is that its uncertainty estimates are very sensitive to the choice of the prior precision hyperparameter (Daxberger et al., 2021b). Our

work looks at the model evidence maximisation method for choosing this hyperparameters, as used in the seminal work of Mackay (1992a). In contrast with often used cross-validation, evidence maximisation reduces model selection to an (often convex) optimisation problem, and can scale to a large number of hyperparameters.

The methods studied in this chapter differ from those of Mackay (1992a) in that we deal with the fully *post-hoc setting*. In modern settings, retraining our NN every time we update the hyperparameters is prohibitively expensive. Thus, we work with a pre-trained NN and do not re-train it once the hyperparameters have been updated. This chapter also differs from the recent body of work of Immer et al. (2021a, 2023a, 2022), since the latter focuses on the *online* setting, where the NN is trained and the hyperparameters are optimised concurrently. We consider the post-hoc setting to be the one of most general interest, since it ensures compatibility with existing and future deep learning training techniques.

Our contributions, presented after a review of the necessary preliminaries in Section 5.1, are the identification of certain incompatibilities between the assumptions underlying the classical linearised Laplace model evidence and modern deep learning methodology, and a number of recommendations on how to adapt the method in light of these. In particular:

- A core assumption of linearised Laplace is that the point of linearisation is a minimum of the training loss. When the neural network is not trained to convergence (and this is almost never done), this does not hold and results in severe deterioration of the model evidence estimate. In Section 5.2, we show that this can be corrected by instead considering the optima of the linearised model's loss, that is solving a quadratic optimisation problem.

- In Section 5.3, we show that for networks with normalisation layers (such as batch norm (Ioffe and Szegedy, 2015)), the linearised Laplace predictive distribution can fail to be well-defined. However, this can be resolved by separately parametrising the prior corresponding to normalised and non-normalised network parameters. We also show that a standard feature-normalisation method, the g-prior (Minka, 2000; Zellner, 1986), resolves this pathology.

We provide both theoretical and, in Section 5.5, empirical justification for both points above. The resulting recommended procedure significantly outperforms a naïve linearised Laplace implementation on a series of standard tasks and a wide range of neural architectures: MLPs, classic CNNs, residual networks with and without normalisation layers, generative autoencoders and transformers.

5.1 Post-hoc linearised neural net hyperparameter selection

We consider the problem of selecting a Gaussian prior precision, hereon also referred to as the regulariser, with the objective of obtaining calibrated linearised Laplace uncertainty estimates. We go on to review the aspects of linearised Laplace that pertain to post-hoc selection of this hyperparameter. We refer the reader to Section 3.3 for a detailed review of linearised Laplace.

Setup and notation

We consider the *post-hoc* setting. We work with a neural network $g\colon \mathcal{V} \times \mathcal{X} \mapsto \mathcal{Y}$ with parameter space $\mathcal{V} \subseteq \mathbb{R}^d$, input space $\mathcal{X}$ and output space $\mathcal{Y} \subseteq \mathbb{R}^c$. We assume access to a pre-trained set of weights $\tilde{v} \in \mathcal{V}$ which we keep fixed throughout the chapter. Additionally, we assume these were obtained by minimising a regularised objective of the form

$$\mathcal{L}_{g,A}(v) = L(g(v, \cdot)) + \|v\|_A^2, \tag{5.1}$$

for $L\colon \mathcal{Y}^{\mathcal{V}\times\mathcal{X}} \mapsto \mathbb{R}_+$ of the form $L(g(v,\cdot)) = \sum_i^n \ell(y_i, g(v, x_i))$ where ℓ is a negative log-likelihood function. We assume any linking functions are absorbed into ℓ. $\|v\|_A^2$ corresponds to the log density of a Gaussian prior over v for some initial value the of the positive-definite prior precision matrix $A \in \mathbb{R}^{d\times d}$. However, we henceforth treat A as a model hyperparameter. Throughout this chapter we use $\cdot$ to denote by a vector-matrix or matrix-matrix product where this may help with clarity.

Linearisation and posterior approximation

The parameter setting $\tilde{v}$ acts as the linearisation point around which we approximate g with the affine function

$$h(w, x) = g(\tilde{v}, x) + \partial_v[g(v, x)](\tilde{v}) \cdot (w - \tilde{v}), \tag{5.2}$$

with parameters $w \in \mathbb{R}^d$. We then approximate the loss function for the linearised model, $\mathcal{L}_{h,A}(w) = L(h(w,\cdot)) + \|w\|_A^2$, with a second order Taylor expansion about $\tilde{v}$. Since $\partial_w h(\tilde{v}, \cdot) = \partial_v g(\tilde{v}, \cdot)$ and, by assumption, $\tilde{v} \in \arg\min_v \mathcal{L}_{g,A}$, we have that $\partial_w \mathcal{L}_{h,A}(\tilde{v}) = \partial_v \mathcal{L}_{g,A}(\tilde{v}) = 0$, and thus the first order term vanishes. This leaves us with the approximation

$$\mathcal{L}_{h,A}(\tilde{v}) + \frac{1}{2}\|w - \tilde{v}\|^2_{\partial_w^2 \mathcal{L}_{h,A}(\tilde{v})}. \tag{5.3}$$

We define an approximate posterior Q by taking its Lebesgue density to be proportional to the exponential of minus this approximate loss. That is, a Gaussian with mean $\tilde{v}$ and covariance $(\partial_w^2 \mathcal{L}_{h,A}(\tilde{v}))^{-1}$. We henceforth adopt the notation of Chapter 2 and Section 3.3, writing the Hessian of $\mathcal{L}_{h,A}$ at $\tilde{v}$ as

$$\partial_w^2 \mathcal{L}_{h,A}(\tilde{v}) = M + A \quad \text{with} \quad M = \partial_w^2 [L(h(w, \cdot))](\tilde{v}), \tag{5.4}$$

and $J(\cdot) = \partial_v g(\tilde{v}, \cdot)$ for the Jacobian of g at $\tilde{v}$. The approximate predictive posterior is given by the GP $h(w, \cdot)$, $w \sim Q$. Since h is affine, this is again Gaussian. Its marginal at a test point $x' \in \mathcal{X}$ is

$$\mathcal{N}(g(w_\star, x'),\ J(x')(M + A)^{-1} J(x')^T). \tag{5.5}$$

Model selection

The predictive posterior $h(w, \cdot)$, $w \sim Q$ corresponds to a GP with covariance kernel $J(\cdot)(M + A)^{-1} J(\cdot')^T$, revealing an explicit dependence on the regulariser A. This parameter significantly affects the predictive posterior variance, but we have no simple method for choosing it *a priori*. We instead follow an empirical-Bayes procedure: we interpret A as the precision of a prior $\Pi = \mathcal{N}(0, A)$ and choose A as that most likely to generate the observed data given the prior linearised model $h(w, \cdot);\ w \sim \Pi$. This yields the objective

$$\mathcal{G}_{\tilde{v}}(A) = -\frac{1}{2} \left[\|\tilde{v}\|_A^2 + \log \det(A^{-1} M + I) \right] + C, \tag{5.6}$$

where C is independent of A. We have made explicit the objective's dependence on the linearisation point, which we assume fixed throughout optimisation of A, with the subscript $_{\tilde{v}}$. Equation (5.6) is called the model evidence. Throughout, we will constrain A to the set of positive diagonal matrices, as in Mackay (1992a). Maximising $\mathcal{G}_{\tilde{v}}$ is a concave optimisation problem.

Discussion: advantages and limitations of linearised Laplace in the modern setting

The posterior predictive mean is fixed to match $g(\tilde{v}, \cdot)$, ignoring that a change in A will almost surely change the modes of $\mathcal{L}_{g,A}$. This choice keeps the NN's predictions unchanged. This is considered an advantage of linearised Laplace over competing Bayesian deep learning methods, which are often forced to compromise the accuracy of their predictive mean for better calibrated uncertainty.

We made a number of assumptions in our derivation. First, that the data-fit term L is convex. This is satisfied by the standard losses used to train neural networks. We also assumed that the true posterior over the NN weights is sharply peaked around its optima such that it can be approximated well by a quadratic expansion and that h is a good approximation to g near the linearisation point. These assumptions we do not question further. We made one further important assumption, that the linearisation point $\tilde{v}$ is a local minimum of $\mathcal{L}_{g,A}$ and thus it is also a minima of the linearised loss $\mathcal{L}_{h,A}$. This final assumption will be the focus of our work.

Since the linearised Laplace method with model-evidence maximisation was first introduced by Mackay (1992a), deep learning training procedures and architectures have changed. Stochastic first order methods are used to minimise the loss function in place of the second order full-batch methods common in classical literature (Amari et al., 2000; LeCun et al., 1996). We often do not use a low value of the loss $\mathcal{L}_{g,A}$ as a stopping criterion, but instead monitor some separate validation metric. Also, normalisation layers are ubiquitous.

Since the derivations of this section assume that we linearise g and expand $\mathcal{L}_{h,A}$ about a local minimum of $\mathcal{L}_{g,A}$ (and thus of $\mathcal{L}_{h,A}$), modern practises pose difficulties for the presented method. The rest of this chapter explores these issues, proposes a modern adaptation of the linearised Laplace method, and discusses some interesting special cases.

5.2 On the choice of posterior mode

We consider a naïve implementation of the linearised Laplace method in the context of modern neural networks as that using the linearisation point $\tilde{v}$ in the expression for the model evidence $\mathcal{G}_{\tilde{v}}$ (5.6), even if this point is known to not be a local minimum of $\mathcal{L}_{g,A}$. We now propose an alternative.

We begin, as before, by linearising g about $\tilde{v}$, the point returned by (possibly stochastic or incomplete) optimisation of the neural network loss $\mathcal{L}_{g,A}$ and constructing the feature expansion $J\colon x \mapsto \partial_v g(\tilde{v}, \cdot)$. Under broad assumptions discussed in Section 5.4.1, this choice means the posterior mean $g(\tilde{v}, \cdot)$ is contained within the linear span of the Jacobian features $J(\cdot)$[1]. This yields credence to the interpretation of the linear model's error bars as uncertainty about the NN output.

[1]When the NN's output layer is linear, $g(\tilde{v}, \cdot)$ is a linear combination of the final layer activations and the Jacobian contains the last layer activations.

We diverge from Section 5.1 in how we approximate $\mathcal{L}_{h,A}$. We start by noting that since $\tilde{v}$ is not a local minimum of $\mathcal{L}_{g,A}$, it is not one of $\mathcal{L}_{h,A}$ either.

Observation 2. For network g with linearisation h about $\tilde{v}$ and a positive definite regulariser A, if $\tilde{v}$ is not a stationary point of $\mathcal{L}_{g,A}$, it is not a local minimum of $\mathcal{L}_{h,A}$.

Derivation Proof of observation 2

Proof. Since $\tilde{v}$ is not a stationary point of $\mathcal{L}_{g,A}$, the gradient $\partial_v \mathcal{L}_{g,A}(\tilde{v})$ is not identically zero. But $\partial_v \mathcal{L}_{g,A}(\tilde{v})$ equal to

$$\sum_i \partial_{\hat{y}_i}[\ell(\hat{y}_i, y_i)](g(\tilde{v}, x_i))\partial_v[g(v, x_i)](\tilde{v}) + \partial_v[\|v\|_A^2](\tilde{v})$$
$$= \sum_i \partial_{\hat{y}_i}[\ell(\hat{y}_i, y_i)](h(\tilde{v}, x_i))\partial_w[h(w, x_i)](\tilde{v}) + \partial_w[\|w\|_A^2](\tilde{v}),$$

which is in turn equal to $\partial_w \mathcal{L}_{h,A}(\tilde{v})$. Since this is thus non-zero, $\tilde{v}$ cannot be a local minimum of $\mathcal{L}_{h,A}$. □

Thus, $\tilde{v}$ is not a suitable point for a quadratic approximation to $\mathcal{L}_{h,A}$ without a first order term. However, for any given A, the loss for the linearised model $\mathcal{L}_{h,A}$ is a convex function of w (L is convex and h is linear in w), and thus has a well defined minimiser; expanding the loss about this minimiser will yield a more faithful approximation to the evidence. Moreover, for each fixed w, $\mathcal{G}_w(A)$ is concave in A, yielding a maximiser. Iteratively minimising the convex $\mathcal{L}_{h,A}(w)$ and maximising the concave $\mathcal{G}_w(A)$ yields a simultaneous stationary point $(w_\star, A_\star)$ satisfying

$$w_\star \in \arg\min_w \mathcal{L}_{h,A_\star}(w) \quad \text{and} \quad A_\star \in \arg\max_A \mathcal{G}_{w_\star}(A).$$

Our adaption performs evidence maximisation with an affine model h where the basis expansion J is fixed. Unlike Mackay (1992a), we do not retrain the neural network. Instead, we re-fit the linear model. Chapter 6 will introduce methods that efficiently implement this iterative optimisation scheme.

In practice, we make one further approximation: rather than evaluating the curvature $\partial_w^2 L(h(w, \cdot))$ afresh at successive modes of $\mathcal{L}_{h,A}$ found during the iterative procedure for computing $(w_\star, A_\star)$, we use the curvature at the linearisation point $M = \partial_w^2[L(h(w, \cdot))](\tilde{v})$ throughout. This avoids the expensive re-computation of the Hessian; experimentally we

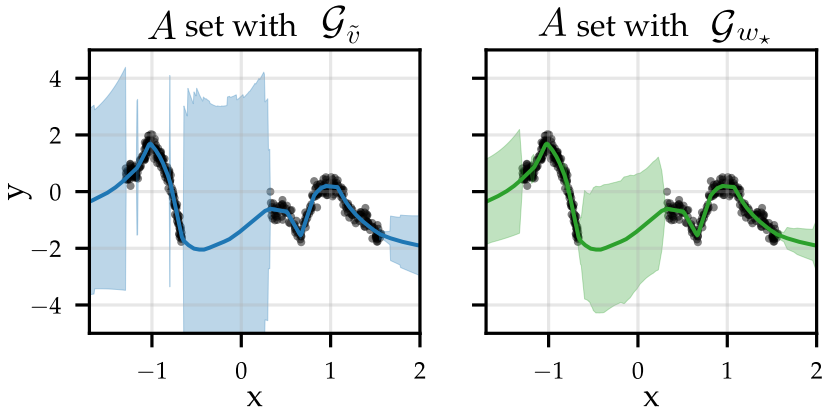


Fig. 5.1 Linearised Laplace predictive mean and std-dev for a 2.6k parameter MLP trained on toy dataset from Antorán et al. (2020). Choosing A with $\mathcal{G}_{\tilde{v}}$ yields error bars larger than the marginal std-dev of the targets. Recommendation 1 (using $\mathcal{G}_{w_\star}$) solves this.

find that this does not affect the results[2] (see Section 5.5.1). The resulting model evidence expression matches that in (5.6), with only the weights featuring in the norm changed,

$$\mathcal{G}_{w_\star}(A) = -\frac{1}{2}\left[\|w_\star\|_A^2 + \log\det(A^{-1}M + I)\right] + C. \tag{5.7}$$

Recommendation 1. *While using the linearisation point $\tilde{v}$ in the construction of the feature expansion J and the Hessian M (as introduced in Section 5.1), find a joint optimum $(w_\star, A_\star)$ for the feature-linear model and employ these to construct the corresponding model evidence $\mathcal{G}_{w_\star}$ (equation (5.7)) and to compute the predictive variance (equation (5.5)).*

We thus recommend employing a posterior distribution for h of the same form as given in (5.5), but with prior precision $A_\star$. We do not recommend using the mean predictions of the tangent linear model $h(w_\star, \cdot)$ as the posterior mean function since this introduces additional computational load while empirically providing little to no benefit. We verified this across a range of tasks, including image classification and tomographic image reconstruction. This is illustrated for a 1d toy problem in Figure 5.2. Here, the linearised model's mean resembles the NN's mean but is less smooth. We attribute this non-smoothness to the inclusion of a linear dependence on ReLU features from network layers near the input.

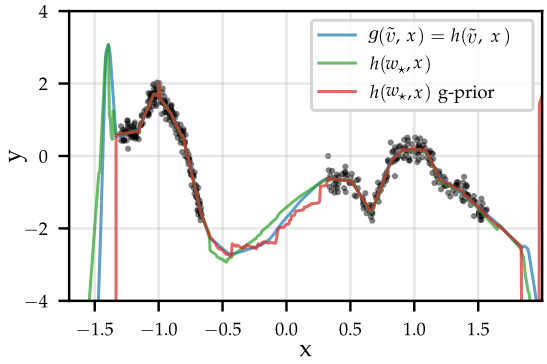


Fig. 5.2 Comparison of the predictive means a a 2.6k parameter MLP trained on toy dataset from Antorán et al. (2020) (blue) with the posterior mean of its tangent linear model with an isotropic Gaussian prior (green) and the posterior mean of the tangent linear model with the diagonal g-prior, introduced in Section 5.3.2 (red).

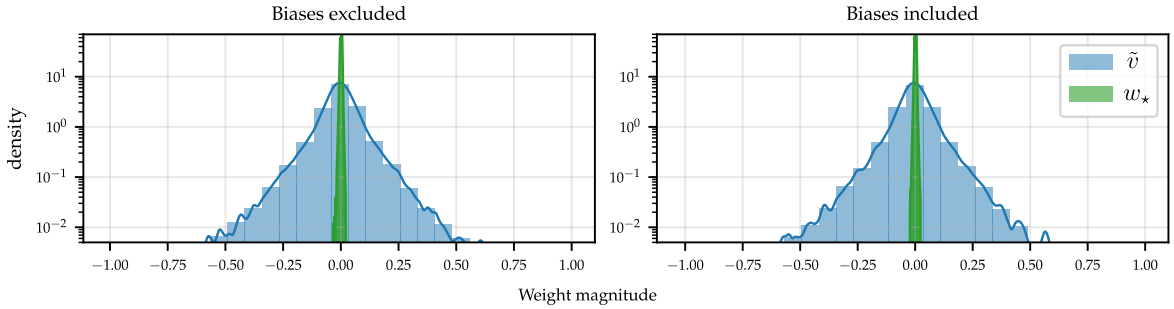


Fig. 5.3 Histograms of the individual entries of $\tilde{v}$ and $w_\star$ for the models in the bias exclusion experiment of Section 5.5.1. We use a $d = 46$k ResCNN described in Section 5.5 and train it on MNIST.

Demonstration: 1d regression and a simple CNN

Figure 5.1 shows how choosing the prior precision with the evidence objective that contains the linearisation point $\tilde{v}$ results in uncertainty overestimation; the predictive distribution's marginal standard deviation is much larger than the marginal standard deviation of the targets. This is resolved by applying recommendation 1. We further explore the differences between the norm of the linear model MAP $w_\star$ and the linearisation point $\tilde{v}$ by, in Figure 5.3, plotting the histogram for both given a small ResNet-style CNN described in Section 5.5. The linear model weights present a much narrower distribution around 0. This is commensurate with their use in the model evidence resulting in larger prior precisions and thus smaller errorbars.

[2]The Hessian depends on the linear model weights w only trough the predictions made by the linear model. If our pre-trained NN is well-fit to the data, we do not expect the linearised NN's MAP predictions to differ much from the NN prediction at the linearisation point.

The plot also ablates whether considering model biases in the linearisation makes a difference to this recommendation, and it does not.

5.3 Linearised Laplace with normalised networks

We now study linearised Laplace in the presence of scale-invariance introduced by normalisation layers. For this, we put forth the following formalism:

Definition 3 (Normalised networks)**.** We say that a set of networks $\mathfrak{G} \subset \mathcal{Y}^{\mathcal{V}\times\mathcal{X}}$ is normalised if $\mathcal{V}$ can be written as a *direct sum* $\mathcal{V}' \oplus \mathcal{V}''$, with $\mathcal{V}''$ non-empty, such that for all networks $g \in \mathfrak{G}$ and parameters $v' + v'' \in \mathcal{V}' \oplus \mathcal{V}''$,

$$g(v' + v'', \cdot) = g(v' + \mathfrak{c}v'', \cdot)$$

for all $\mathfrak{c} \in \mathbb{R}_+$.

Throughout, we write v', v'' to denote the respective projections onto $\mathcal{V}', \mathcal{V}''$ of a parameter $v \in \mathcal{V}$; for ease of notation, we will assume these projections are aligned with a standard basis on $\mathcal{V}$. That is, we write our parameter vectors as the sum of $\mathfrak{c}v''$, which is only non-zero for normalised weights and g is invariant to $\mathfrak{c}$, and v', for which the opposite holds.

Remark Example to illustrate how definition 3

Consider an MLP $g : \mathcal{V} \times \mathcal{X} \to \mathcal{Y}$ with a single input dimension $\mathcal{X} = \mathbb{R}$, a single output dimension $\mathcal{Y} = \mathbb{R}$, single hidden layer and 2 hidden units $\mathcal{V} = \mathbb{R}^4$. We apply layer norm after the input layer parameters. The model parameters are $v = [v_1, v_2, v_3, v_4] \in \mathcal{V}$ with v_1, v_2 belonging to the input layer and v_3, v_4 to the readout layer. We assume there are no biases without loss of generality.
Denoting the outputs of the first parameter layer $\mathfrak{a} = [v_1 x, v_2 x]$, layer norm applies the function

$$\frac{\mathfrak{a} - \mathbb{E}[\mathfrak{a}]}{\sqrt{\text{Var}(\mathfrak{a})}}\mathfrak{b} + \mathfrak{e} \quad \text{with} \quad \mathbb{E}[\mathfrak{a}] = 0.5v_1 x + 0.5v_2 x$$
$$\text{and} \quad \text{Var}(\mathfrak{a}) = 0.5(v_1 x - \mathbb{E}[\mathfrak{a}])^2 + 0.5(v_2 x - \mathbb{E}[a])^2$$

for $\mathfrak{b} \in \mathbb{R}$, $\mathfrak{e} \in \mathbb{R}$. Now take $\mathfrak{c} \in \mathbb{R}_+$ to see that the output of the layernorm layer is invariant to scaling the input layer parameters by $\mathfrak{c}$

$$\frac{\mathfrak{c}\mathfrak{a} - \mathbb{E}[\mathfrak{c}\mathfrak{a}]}{\sqrt{\text{Var}(\mathfrak{c}\mathfrak{a})}} = \frac{\mathfrak{c}(\mathfrak{a} - \mathbb{E}[\mathfrak{a}])}{\mathfrak{c}\sqrt{\text{Var}(\mathfrak{a})}} = \frac{\mathfrak{a} - \mathbb{E}[\mathfrak{a}]}{\sqrt{\text{Var}(\mathfrak{a})}}.$$

Thus, we have $g([v_1, v_2, v_3, v_4], \cdot) = g([\mathfrak{c}v_1, \mathfrak{c}v_2, v_3, v_4], \cdot)$.
Now let $\mathcal{V}$ be the result of the internal *direct sum* $\mathcal{V} = \mathcal{V}' \oplus \mathcal{V}''$, and v', v'' be projections of v onto the subspaces $\mathcal{V}'$ & $\mathcal{V}''$, respectively, so that $v = v' + v''$. The operator $+$ is defined as the vector sum, as usual. In the simplest case where $\mathcal{V}'$ and $\mathcal{V}''$ are aligned with the standard basis, this corresponds to vectors in $\mathcal{V}'$ having zero valued entries in the place of parameters to which normalisation is applied, $v' = [0, 0, v_3, v_4]$. Vectors in $\mathcal{V}''$ are non-zero for normalised parameters, $v'' = [v_1, v_2, 0, 0]$. Finally, we write the property of interest

$$g(v' + \mathfrak{c}v'', \cdot) = g(v' + v'', \cdot).$$

Our formalism requires only that a single group of normalised parameters $\mathcal{V}''$ exists. However, by applying the definition repeatedly, introducing a separate scaling constant per layer, we encompass networks with any number of normalisation layers, and all our results extend to this case. This formalism can be used to model the scale-invariant effect of layer norm (Ba et al., 2016), group norm (Wu and He, 2020) or batch norm (Ioffe and Szegedy, 2015), and even some so-called normalisation-free methods (Brock et al., 2021a,b). However, it is worth noting that each of these normalisation strategies introduce additional effects that are not of interest to this chapter and are deliberately not described by our formalism.

Our focus on normalised networks is motivated by the following observation:

Proposition 4. *For any normalised network g and positive definite matrix A, the loss $\mathcal{L}_{g,A}$ has no local minima.*

To see this, note that the data term fit $L(g(v' + \mathfrak{c}v'', \cdot))$ is invariant to the choice of $\mathfrak{c} > 0$, but we can always decrease the prior term $\|v' + \mathfrak{c}v''\|_A^2$ by decreasing $\mathfrak{c}$. Since $\mathfrak{c} \in \mathbb{R}_+$ has no minimal value, $\mathcal{L}_{g,A}(v' + \mathfrak{c}v'') = L(g(v' + \mathfrak{c}v'', \cdot)) + \|v' + \mathfrak{c}v''\|_A^2$ has no local minima. This is illustrated in Figure 5.4.

As in Section 5.2, minimisers of the linear loss $\mathcal{L}_{h,A}$ remain well-defined (the loss remains strictly convex). However, in this case, $\tilde{v}$ cannot minimise $\mathcal{L}_{h,A}$: the linearisation point

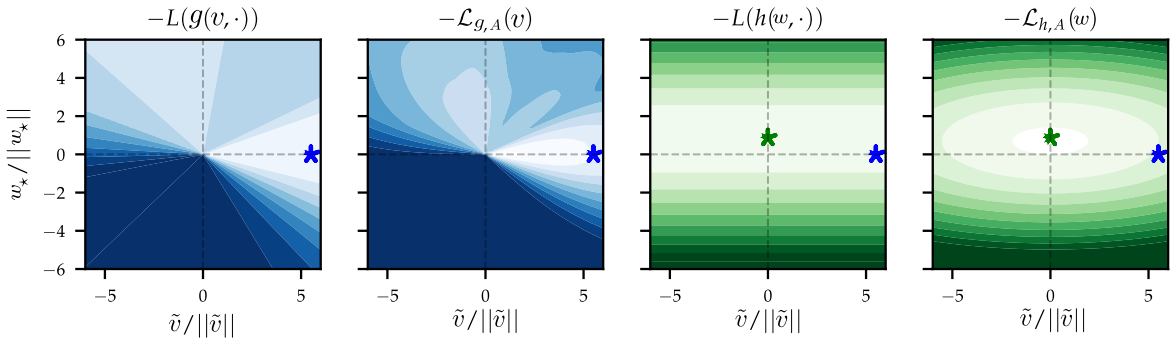


Fig. 5.4 Log likelihood $L(g(v,\cdot))$ (left) and log posterior $\mathcal{L}_{g,A}$ (left middle) density for an MLP with layer norm, both plotted as functions of a 2d slice of the input layer weights. The horizontal axis corresponds to the direction of $\tilde{v}$ while the vertical to $w_\star$. The linearisation point found with SGD $\tilde{v}$ (★) is not an optima of $\mathcal{L}_{g,A}$. We can always increase the value of $\mathcal{L}_{g,A}$ by moving towards the origin along the horizontal axis, without changing the likelihood. $\tilde{v}$ is not an optima of the linear model's Log likelihood $L(h(w,\cdot))$ (middle right) or log posterior $\mathcal{L}_{g,A}$ (right) either. The linear model log posterior $\mathcal{L}_{h,A}$ is convex and optimised by $w_\star$ (★).

minimises $\mathcal{L}_{h,A}$ only if it minimises $\mathcal{L}_{g,A}$ (recall observation 2), and this is now impossible! To correct this, from hereon we follow recommendation 1.

An even larger concern raised by proposition 4 is that the linearisation point is identified only up to the scaling $\mathfrak{c}$ of the normalised parameters $\tilde{v}''$. Since $\mathfrak{c}$ is arbitrary, and does not affect the predictions of the neural network (by definition), it ought not affect the predictive variance returned by the linearised Laplace method. However, due to scaling of the Jacobian features with $\mathfrak{c}$ which we go on to show in the following section, in general, it does. See Figure 5.5 for a demonstration of this.

5.3.1 The layerwise prior

As shown by the following proposition, it suffices to regularise linear model weights corresponding to the normalised parameters $w'' \in \mathcal{V}''$ separately from $w' \in \mathcal{V}'$, and choose both regularisation strengths with the model evidence (5.7), to recover a unique predictive posterior independent of $\mathfrak{c}$.

Proposition 5. *For normalised neural networks, using a regulariser of the form* $\|w'\|^2_{A'} + \|w''\|^2_{A''}$ *with* A' *and* A'' *parametrised independently and chosen according to recommendation 1, the predictive posterior* $h(w,\cdot)$, $w \sim Q$ *induced by a linearisation point* $\tilde{v}' + \mathfrak{c}\tilde{v}''$ *is independent of the choice of* $\mathfrak{c} > 0$.

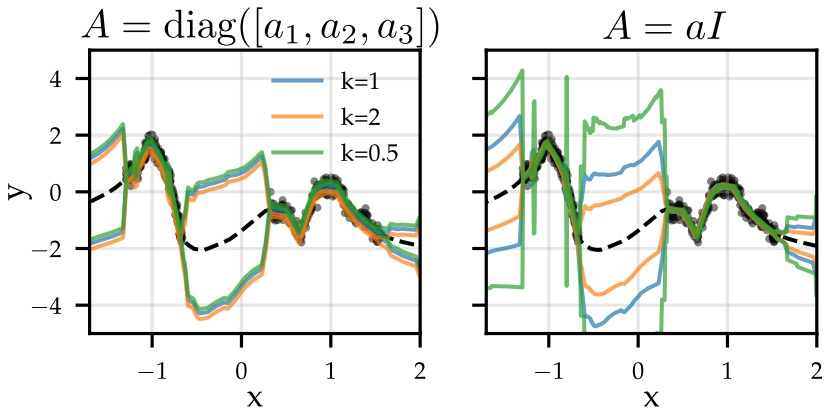


Fig. 5.5 For a normalised MLP with an isotropic prior precision, modifying the scale of the normalised weights $\tilde{v}''$ in the linearisation point changes the error bars after hyper-parameter optimisation (right). Incorporating recommendation 2 fixes the issue (left).

Briefly, the result follows because the Jacobian entries corresponding to weights $\mathfrak{c}v''$ scale with $\mathfrak{c}^{-1}$. This is illustrated in Figure 5.4 (leftmost plot), where as we move further from the origin, weight settings of equal likelihood $L(g(v, \cdot))$ move further from each other. Given an un-scaled reference solution $(w_\star, A_\star)$, as we vary $\mathfrak{c}$ in $\mathfrak{c}\tilde{v}''$, the linear model weights and prior precisions that simultaneously optimise $\mathcal{L}_{h,A_\star}$ and $\mathcal{G}_{w_\star}$ scale as $(w'_\star, \mathfrak{c}w''_\star)$, and $(A'_\star, \mathfrak{c}^{-2}A''_\star)$ respectively. These scalings cancel each other in the predictive posterior, which remains invariant. When A'' can not change independently of A', this cancellation does not occur. An empirical demonstration is provided in Figure 5.5. We now present the full proof.

Derivation Proof of proposition 5

Notation Consider a linearisation point $\tilde{v}' + \tilde{v}''$, with corresponding linearised function h, basis function J and Hessian M. For $\mathfrak{c} > 0$, $h_\mathfrak{c}, J_\mathfrak{c}, M_\mathfrak{c}$ denote these quantities corresponding to a linearisation point $\tilde{v}_\mathfrak{c} := \tilde{v}' + \mathfrak{c}\tilde{v}''$. Moreover, we write

$$J = \begin{bmatrix} J' \\ J'' \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} M' & X^T \\ X & M'' \end{bmatrix}$$

for the sub-entries of J and M with dependencies on v' and v'' respectively, with X containing cross-terms. We refer to sub-entries of $J_\mathfrak{c}$ and $M_\mathfrak{c}$ in the same manner. With notation in place, we have the following scaling result:

Lemma 6. *Let g be a normalised network and consider two alternative linearisation points $\tilde{v} = \tilde{v}' + \tilde{v}''$ and $\tilde{v}_\mathfrak{c} = \tilde{v}' + \mathfrak{c}\tilde{v}''$ for some $\mathfrak{c} > 0$. Then,*

$$\begin{bmatrix} J'_\mathfrak{c} \\ \mathfrak{c}J''_\mathfrak{c} \end{bmatrix} = J \quad \textit{and} \quad \begin{bmatrix} M'_\mathfrak{c} & \mathfrak{c}X_\mathfrak{c}^T \\ \mathfrak{c}X_\mathfrak{c} & \mathfrak{c}^2 M''_\mathfrak{c} \end{bmatrix} = M.$$

Moreover, for all $w \in \mathcal{V}$, $h_\mathfrak{c}(w' + \mathfrak{c}w'', \cdot) = h(w' + w'', \cdot)$.

Proof. First, we consider $J'_\mathfrak{c}$ and $J''_\mathfrak{c}$. For $J'_\mathfrak{c}$, take any $\tilde{v}' \in \mathcal{V}'$ and consider the directional derivative $D_{v'}g(\tilde{v}' + \mathfrak{c}\tilde{v}'')$. From the limit definition,

$$\begin{aligned} D_{v'}g(\tilde{v}' + \mathfrak{c}\tilde{v}'', \cdot) &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[g((\tilde{v}' + \delta v' + \mathfrak{c}\tilde{v}''), \cdot) - g((\tilde{v}' + \mathfrak{c}\tilde{v}''), \cdot) \right] \\ &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[g((\tilde{v}' + \delta v' + \tilde{v}''), \cdot) - g((\tilde{v}' + \tilde{v}''), \cdot) \right] \\ &= D_{v'}g(\tilde{v}' + \tilde{v}'', \cdot). \end{aligned}$$

From the Jacobian-product definition, we have $J'_\mathfrak{c} \cdot v' = J' \cdot v'$. Since $v' \in \mathcal{V}'$ was arbitrary and we are working on a finite-dimensional Euclidean space, this shows $J'_\mathfrak{c} = J'$. For $J''_\mathfrak{c}$, consider $D_{v''}g(\tilde{v}' + \mathfrak{c}\tilde{v}'')$ for $v'' \in \mathcal{V}''$ arbitrary. We have

$$\begin{aligned} D_{v''}g(\tilde{v}' + \mathfrak{c}\tilde{v}'', \cdot) &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[g((\tilde{v}' + \mathfrak{c}\tilde{v}'' + \delta v''), \cdot) - g((\tilde{v}' + \mathfrak{c}\tilde{v}''), \cdot) \right] \\ &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[g(\tilde{v}' + \tilde{v}'' + \frac{\delta}{\mathfrak{c}} v''), \cdot) - g((\tilde{v}' + \tilde{v}''), \cdot) \right] \\ &= \frac{1}{\mathfrak{c}} \lim_{\delta' \downarrow 0} \frac{1}{\delta'} \left[g((\tilde{v}' + \tilde{v}'' + \delta' v''), \cdot) - g((\tilde{v}' + \tilde{v}''), \cdot) \right] \\ &= \frac{1}{\mathfrak{c}} D_{v''}g(\tilde{v}' + \tilde{v}'', \cdot). \end{aligned}$$

Repeating the same argument as for $J'_\mathfrak{c}$, we obtain $J''_\mathfrak{c} = \frac{1}{\mathfrak{c}} J''$.
Now, we look at the scaling of $h_\mathfrak{c}$. By definition, using that g is normalised and the previously derived scaling for $J_\mathfrak{c}$,

$$\begin{aligned} h_\mathfrak{c}(w' + \mathfrak{c}w'', \cdot) &= g(\tilde{v}' + \mathfrak{c}\tilde{v}'', \cdot) + J'_\mathfrak{c}(w' - \tilde{v}') + J''_\mathfrak{c}(\mathfrak{c}w'' - \mathfrak{c}\tilde{v}'') \\ &= g(\tilde{v}' + \tilde{v}'', \cdot) + J'(w' - \tilde{v}') + J''(w'' - \tilde{v}'') \\ &= h(w' + w'', \cdot), \end{aligned}$$

which is the claimed result.

For $M_{\mathfrak{c}}$, we examine it entry-wise. We have,

$$\begin{aligned}[M_{\mathfrak{c}}]_{mn} &= \partial_{w_m}\partial_{w_n}[L(h_{\mathfrak{c}}(w,\cdot))](\tilde{v}_{\mathfrak{c}}) \\ &= \sum_i \partial_{w_m}[h_{\mathfrak{c}}(w,x_i)](\tilde{v}_{\mathfrak{c}}) \cdot \partial^2_{\hat{y}_i}[\ell(\hat{y}_i,y_i)](h_{\mathfrak{c}}(\tilde{v}_{\mathfrak{c}},x_i)) \cdot \partial_{w_n}[h_{\mathfrak{c}}(w,x_i)](\tilde{v}_{\mathfrak{c}}) \\ &\qquad + \sum_j \partial_{\hat{y}_i}[\ell(\hat{y}_i,y_i)](h_{\mathfrak{c}}(\tilde{v}_{\mathfrak{c}},x_i)) \cdot \partial_{w_m}\partial_{w_n}[h_{\mathfrak{c}}(w,x_j)](\tilde{v}_{\mathfrak{c}}).\end{aligned}$$

Now since $h_{\mathfrak{c}}$ is affine, it has no curvature and thus $\partial_{w_m}\partial_{w_n}h_{\mathfrak{c}}(w,x)$ is identically zero for all $w \in \mathcal{V}$ and $x \in \mathcal{X}$. With that, the second term in the sum vanishes. For the first sum, consider the *middle term*, the curvature of the negative log-likelihood function, and use $h_{\mathfrak{c}}(w' + \mathfrak{c}w'', \cdot) = h(w' + w'', \cdot)$ to see that it is invariant to $\mathfrak{c}$. Finally, note that $\partial_{w_m}h_{\mathfrak{c}}$ and $\partial_{w_n}h_{\mathfrak{c}}$ are entries of $J_{\mathfrak{c}}$ and inherit scaling from therein. Specifically, if both w_m and w_n belong to $\mathcal{V}''$, we obtain $\mathfrak{c}^2$ scaling; if just one belongs to $\mathcal{V}''$, we get $\mathfrak{c}$ scaling, and otherwise we obtain constant scaling. This completes the result for $M_{\mathfrak{c}}$. □

We now turn to how the optimal weights and regularisation parameters scale with the parameter $\mathfrak{c}$.

Lemma 7. *For $\mathfrak{c} > 0$, let $h_{\mathfrak{c}}$ be a linearisation of a normalised network g about $\tilde{v}' + \mathfrak{c}\tilde{v}''$. Then $(w_{\mathfrak{c}}, A_{\mathfrak{c}})$ are an optima of the resulting objectives $(\mathcal{L}_{h_{\mathfrak{c}},A_{\mathfrak{c}}}, \mathcal{G}_{w_{\mathfrak{c}}})$ respectively if and only if they are of the form*

$$(w_{\mathfrak{c}}, A_{\mathfrak{c}}) = (w'_\star + \mathfrak{c}w''_\star,\ A'_\star + \mathfrak{c}^{-2}A''_\star)$$

where $(w_\star, A_\star)$ are optima of $(\mathcal{L}_{h,A_\star}, \mathcal{G}_{w_\star})$ with h a linearisation of g about $\tilde{v}' + \tilde{v}''$.

Proof. To prove the result, we will show that $\mathcal{L}_{h_{\mathfrak{c}},A_{\mathfrak{c}}}(w' + \mathfrak{c}w'') = \mathcal{L}_{h,A_\star}(w' + w'')$ for all $w' + w'' \in \mathcal{V}$ and $\mathcal{G}_{w_{\mathfrak{c}}}(A' + \mathfrak{c}^{-2}A'') = \mathcal{G}_{w_\star}(A' + A'')$ for all strictly diagonal positive matrices A', A'' of compatible sizes. Then, the result follows by noting that for $\mathfrak{c} > 0$ fixed, the mappings $w' + w'' \mapsto w' + \mathfrak{c}w''$ and $A' + A'' \mapsto A' + \mathfrak{c}^{-2}A''$ are bijections. Consider the objective $\mathcal{L}_{h_{\mathfrak{c}},A_{\mathfrak{c}}}$. By definition, $\mathcal{L}_{h_{\mathfrak{c}},A_{\mathfrak{c}}}(w' + \mathfrak{c}w'')$ is given by

$$\begin{aligned}&L(h_{\mathfrak{c}}(w'+\mathfrak{c}w'',\cdot)) + \|w'\|^2_{A'_{\mathfrak{c}}} + \|\mathfrak{c}w''\|^2_{A''_{\mathfrak{c}}} \\ &\qquad = L(h(w'+w'',\cdot)) + \|w'\|^2_{A'_\star} + \|w''\|^2_{A''_\star},\end{aligned}$$

where the equality follows by lemma 6 and the definition of $A_{\mathfrak{c}}$. The bottom expression is equal to $\mathcal{L}_{h,A_\star}(w' + w'')$ proving the equality for the loss term.

Consider the objective $\mathcal{G}_{w_\mathfrak{c}}$. For our claim, we need to show that

$$\|w_\mathfrak{c}\|^2_{A'+\mathfrak{c}^{-2}A''} + \log \frac{\det(M_\mathfrak{c} + A' + \mathfrak{c}^{-2}A'')}{\det(A' + \mathfrak{c}^{-2}A'')} = \|w_\star\|^2_{A'+A''} + \log \frac{\det(M + A' + A'')}{\det(A' + A'')}.$$

The equality $\|w_\mathfrak{c}\|_{A'+\mathfrak{c}^{-2}A''} = \|w_\star\|_{A'+A''}$ holds trivially. We now show equality of the determinants. Let d', d'' denote the dimensions of $\mathcal{V}'$ and $\mathcal{V}''$ respectively. By the Schur determinant lemma, the numerator $\det(M_\mathfrak{c} + A' + \mathfrak{c}^{-2}A'')$ is equal to

$$\det(M''_\mathfrak{c} + \frac{A''_{d':}}{\mathfrak{c}^2}) \det(M'_\mathfrak{c} + A'_{:d'} - X(M''_\mathfrak{c} + \frac{A''_{d':}}{\mathfrak{c}^2})^{-1}X^T),$$

where $A'_{:d'} = [A'_{ij} \colon i, j \leq d']$ and $A''_{d':}$ is defined similarly. Using lemma 6, $\det(M''_\mathfrak{c} + \frac{A''_{d':}}{\mathfrak{c}^2}) = (\frac{1}{\mathfrak{c}^2})^{d''}\det(M'' + A''_{d':})$. Expanding the Schur complement term and using lemma 6 shows that it is independent of $\mathfrak{c}$. In turn, the denominator is given by

$$\begin{aligned}\det(A' + \mathfrak{c}^{-2}A'') &= (\frac{1}{\mathfrak{c}^2})^{d''}\det(A'_{:d'})\det(A''_{d':}) \\ &= (\frac{1}{\mathfrak{c}^2})^{d''}\det(A' + A''),\end{aligned}$$

The $(\frac{1}{\mathfrak{c}^2})^{d''}$ terms in the numerator and denominator cancel, yielding the claim. □

Proof of proposition 5. Using lemma 6 and lemma 7 and the notation defined therein,

$$\|J\|^2_{(M+A_\star)^{-1}} = \|J_\mathfrak{c}\|^2_{(M_\mathfrak{c}+A_\mathfrak{c})^{-1}}.$$

Thus the errorbars induced by linearising about $\tilde{v}' + \tilde{v}''$ and $\tilde{v}' + \mathfrak{c}\tilde{v}''$ are equal for all $\mathfrak{c} > 0$. □

We note that proposition 5 holds even when $A_\star$ is found by evaluating the Hessian at the optima of the linear model loss instead of $w_\star$, instead of linearisation point $\tilde{v}$—the latter is our suggestion in Section 5.2. This is because $w''_\star$ scales with $\tilde{v}''$ (lemma 7).

By induction, proposition 5 applies to networks with multiple normalisation layers. Note that the proof of the results required for proposition 5 depends crucially on being able to scale $A''_\mathfrak{c}$ with $\mathfrak{c}$ while keeping $A'_\mathfrak{c}$ fixed. This motivates our recommendation:

Recommendation 2. *When using the linearised Laplace method with a normalised network, use an independent regulariser for each normalised parameter group present.*

An example of a suitable regulariser for a network with normalised parameter groups $v^{(1)}, v^{(2)}, \ldots, v^{(L)}$ and non-normalised parameters v' would be

$$a'\|w'\|^2 + a_1\|w^{(1)}\|^2 + a_2\|w^{(2)}\|^2 + \ldots + a_L\|w^{(L)}\|^2$$

for independent parameters $a', a_1, a_2, \ldots, a_L > 0$ and $w^{(1)}, w^{(2)}, \ldots, w^{(L)}$ referring to the linear model weights corresponding to the NN weights in each normalised parameter group. Usually, this involves setting independent priors for each layer of the network.

5.3.2 The diagonal g-prior

We now present a different class of diagonal prior which exploits the scaling of the likelihood curvature with the linearisation point (lemma 6) to resolve the issue of scale indeterminacy in the predictive posterior.

Proposition 8. *For normalised neural networks, using a regulariser of the form $\|w\|_A^2$ with*

$$A = a \operatorname{diag} M$$

for $a \in \mathbb{R}_+$ and $M = \partial_w^2[L(h(w,\cdot))](\tilde{v})$, the predictive posterior $h(w,\cdot)$, $w \sim Q$ induced by a linearisation point $\tilde{v}' + \mathfrak{c}\tilde{v}''$ is independent of the choice of $\mathfrak{c} > 0$.

Derivation Proof of proposition 8

We adopt the notation used in lemma 6 and lemma 7.

Proof. Let $A = a \operatorname{diag} M$ for a model with linearisation point $\tilde{v}' + \tilde{v}''$. By lemma 6, for a model with linearisation point $\tilde{v}_\mathfrak{c} := \tilde{v}' + \mathfrak{c}\tilde{v}''$, with $\mathfrak{c} > 0$, the corresponding regulariser is

$$A_\mathfrak{c} = a \begin{bmatrix} \operatorname{diag} M' & 0 \\ 0 & \mathfrak{c}^{-2} \operatorname{diag} M'' \end{bmatrix} = a \begin{bmatrix} \operatorname{diag} M'_\mathfrak{c} & 0 \\ 0 & \operatorname{diag} M''_\mathfrak{c} \end{bmatrix}.$$

With that, lemma 6 and lemma 7, we have

$$\|J\|^2_{(M+A)^{-1}} = \|J_\mathfrak{c}\|^2_{(M_\mathfrak{c}+A_\mathfrak{c})^{-1}}.$$

Thus the errorbars induced by linearising about $\tilde{v}' + \tilde{v}''$ and $\tilde{v}' + \mathfrak{c}\tilde{v}''$ are equal for all $\mathfrak{c} > 0$. □

This is a diagonal version of what is known in the literature as the g-prior (Zellner, 1986) or scale-invariant prior (Minka, 2000). It has the advantage over the layer-wise prior of only having one free parameter to learn via the evidence. Additionally, unlike the layerwise prior, the posterior corresponding to the g-prior is invariant to the scale of the linearisation point for any value of the free parameter $a \in \mathbb{R}_+$, not just for the one that maximises the evidence $\mathcal{G}_{w_\star}$. A practical implementation must ensure that no entries of diag M are 0 to preserve positive definiteness in cases where the log-likelihood function is not strictly convex. A further advantage of the diagonal g-prior is that it normalises the scales of the Jacobian entries corresponding to different NN weights, as illustrated in Figure 5.6. For this reason, the diagonal g-prior may, in general, improve the conditioning of the linearised model's loss $\mathcal{L}_{h,A}$. Indeed, this prior is intimately related to the Jacobi preconditioner.

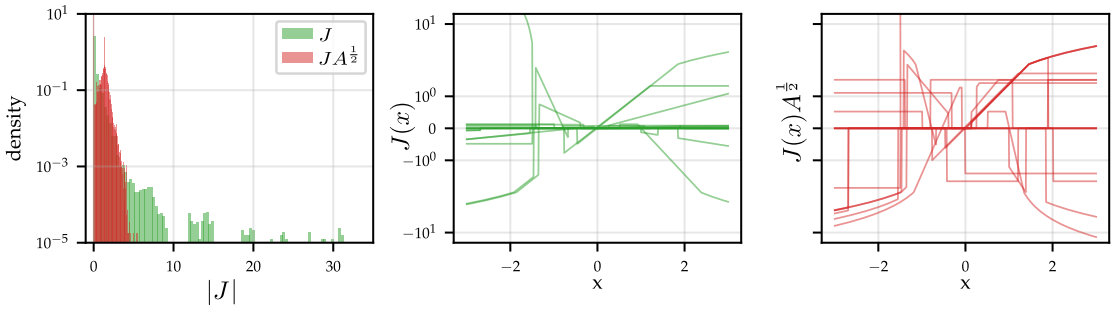


Fig. 5.6 Left: Histogram of the absolute value of Jacobian entries, training data, across model weights and training datapoints. We use the NN depicted in Figure 5.1, with and without g-prior scaling. Middle: 15 randomly chosen Jacobian basis functions. Right: Same functions with g-prior scaling.

Remark The history of the g-prior

The g-prior was originally introduced by Zellner (1986), It consists of a centred Gaussian with covariance matching the inverse of the Fisher information matrix. Resultantly, the g-prior ensures inferences are independent of the units of measurement of the covariates (Minka, 2000). Since then, it has extensively used in the context of model selection for generalised linear models (Baragatti and Pommeret, 2012; Bové and Held, 2011; Liang et al., 2008). In the large-scale setting, we have overcome the computational intractability of the Fisher by diagonalising the g-prior while preserving its scale-invariance property.

5.4 Additional observations and discussion

The above analysis leads to a number of observations and further insights into linearised Laplace. The reader should note that the $\cdot$ notation will be doing some heavy lifting in terms of denoting Jacobian vector products taken such that their dimensions are compatible.

5.4.1 Networks with a dense final layer

We look at networks with a dense linear final layer, a (very general) special case. Letting $\mathfrak{l}$ denote the number of non last layer weights such that $v_{:\mathfrak{l}}$ is the vector of all network parameters but those of the last layer, and $v_{\mathfrak{l}:}$ are the last layer weights, we deal with models of the form

$$g(v, \cdot) = \varphi(v_{:\mathfrak{l}}, \cdot) \cdot v_{\mathfrak{l}:}, \tag{5.8}$$

where $\varphi(v_{:\mathfrak{l}}, \cdot)$ is the output of the penultimate layer. The derivative of the neural network with respect to the dense final layer weights is

$$\partial_{v_{\mathfrak{l}:}} g(v, \cdot) = \varphi(v_{:\mathfrak{l}}, \cdot),$$

and thus the final layer activations $\varphi(v_{:\mathfrak{l}}, \cdot)$ are contained within the Jacobian matrix. Consequently, the neural network output $g(v, \cdot)$ is always contained in the linear span of the Jacobian basis. This motivates recommendation 1, where we argue for the use of $\tilde{v}$ for network linearisation, as it allows for an easy linear model error-bars interpretation for the resulting uncertainty.

Also, the form of the linearised model h simplifies in the dense final layer case when the network is fully normalised. Here d'', the dimension of $\mathcal{V}''$, matches $\mathfrak{l}$, and thus we can write $g(v' + v'', \cdot) = \varphi(v''_{:\mathfrak{l}}, \cdot) \cdot v'_{\mathfrak{l}:}$. The derivative of φ in the direction of the linearisation point $\tilde{v}''_{:\mathfrak{l}}$ is zero

$$\partial_{v''_{:\mathfrak{l}}} \varphi(\tilde{v}''_{:\mathfrak{l}}, \cdot) \cdot \tilde{v}''_{:\mathfrak{l}} = 0. \tag{5.9}$$

Thus cancellation occurs in (5.2), as

$$\begin{aligned} h(w, \cdot) &= \varphi(v_{:\mathfrak{l}}, \cdot) \cdot v_{\mathfrak{l}:} + \partial_v g(\tilde{v}, \cdot) \cdot (w - \tilde{v}) \\ &= \varphi(v_{:\mathfrak{l}}, \cdot) \cdot v_{\mathfrak{l}:} + \partial_v g(\tilde{v}, \cdot) \cdot w - \partial_{v_{:\mathfrak{l}}} g(\tilde{v}, \cdot) \cdot v_{:\mathfrak{l}} - \partial_{v_{\mathfrak{l}:}} g(\tilde{v}, \cdot) \cdot v_{\mathfrak{l}:} \\ &= \varphi(v_{:\mathfrak{l}}, \cdot) \cdot v_{\mathfrak{l}:} + \partial_v g(\tilde{v}, \cdot) \cdot w - \partial_{v''_{:\mathfrak{l}}} g(\tilde{v}, \cdot) \cdot v''_{:\mathfrak{l}} - \varphi(v_{:\mathfrak{l}}, \cdot) \cdot v_{\mathfrak{l}:} \\ &= J(\cdot) w, \quad w \sim Q. \end{aligned} \tag{5.10}$$

That is, a linear model based on the features $J(\cdot) = \partial_v [g(v, \cdot)](\tilde{v})$. This removes implementation complications that would stem from considering the zeroth order term in the affine linear model when it comes to finding the MAP of the linearised model $w_\star \in \arg\min_{w \in \mathbb{R}^d} \mathcal{L}_{h,A}(w)$.

Derivation Jacobian null space of fully normalised networks
Consider a normalised network g, the linearisation point $\tilde{v}' + \tilde{v}''$, and the directional derivative with respect to parameters v'' in the direction of $\tilde{v}''$, denoted $D_{\tilde{v}''} g(\tilde{v}' + \tilde{v}'', \cdot)$. On one hand, this is just the partial derivative of g with respect to v'' evaluated at $\tilde{v}''$ and projected onto $\tilde{v}$, and thus $D_{\tilde{v}''} g(\tilde{v}' + c\tilde{v}'', \cdot) = \partial_{v''} g(\tilde{v}, \cdot) \cdot \tilde{v}$. On the other hand, from the limit definition of the directional derivative,

$$\begin{aligned} D_{\tilde{v}''} g(\tilde{v}' + c\tilde{v}'') &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[g(\tilde{v}' + (\delta + c)\, \tilde{v}'', \cdot) - g(\tilde{v}' + c\tilde{v}'', \cdot) \right] \\ &= 0, \end{aligned}$$

and thus $\partial_{v''} g(\tilde{v}, \cdot) \cdot \tilde{v} = 0$.
The quantity appearing in equation (5.9) is $\partial_{v''_{:\mathrm{I}}} \varphi(\tilde{v}''_{:\mathrm{I}}, \cdot) \cdot \tilde{v}''_{:\mathrm{I}}$. We now observe that for a fully normalised network, each of the outputs of the penultimate layer $[\varphi(v_{:\mathrm{I}}, \cdot)]_i$, with the output dimension being indexed by i, is a fully normalised network (definition 3) in its own right. Hence, we can apply the same reasoning as above to see that $\partial_{v''_{:\mathrm{I}}} \varphi(\tilde{v}''_{:\mathrm{I}}, \cdot) \cdot \tilde{v}''_{:\mathrm{I}} = 0$.

5.4.2 Optimising linearised networks

Our adapted linearised Laplace method requires identifying the joint stationary point $(w_\star, A_\star)$. In general, this does not admit a closed-form solution. Instead, we alternate gradient-based optimisation of $\mathcal{L}_{h,A}$ and $\mathcal{G}_v$. For normalised networks with dense output layers, implementing the simplified linear model (5.10) directly yields faster and more stable optimisation. Obtaining the gradients of $\mathcal{G}_v$ involves computing Hessian log-determinants, which in turn requires approximations in the context of large networks. In this chapter's experiments (Section 5.5), we will rely on the KFAC (Martens and Grosse, 2015) approximation for this. In Chapter 6, we will introduce a more accurate sample-based approximation. We go on to provide a derivation for the gradient of $\mathcal{L}_{h,A}$, algorithm 2 and discuss implementation trade-offs.

The linear model loss gradient

We now discuss the optimisation of the loss for the predictor $h(w, \cdot) = J(\cdot) \cdot w$ where $w \in \mathcal{V}$ is the linear model's parameter vector. This corresponds to fully normalised networks with a dense final layer. We note that the procedure for the non-simplified Taylor expanded model $g(\tilde{v}, \cdot) + J(\cdot) \cdot (w - \tilde{v})$ is analogous, but the targets are shifted to be $Y - g(\tilde{v}, \cdot) + \Phi\tilde{v}$. Here, we denote NN Jacobians as $J(\cdot) = \partial_v g(\tilde{v}, \cdot) \in \mathbb{R}^{c \times d}$, we stack then across train points to produce the design matrix $\Phi \in \mathbb{R}^{nc \times d}$, and c is the output dimensionality $|\mathcal{Y}|$.

We wish to optimise w according to the objective $\mathcal{L}_{h,A}(w) = L(h(w, \cdot)) + \|w\|_A^2$. We adopt a first order gradient-based approach. We first consider the gradient of $L(h(w, \cdot)) = \sum_i \ell(J(x_i) \cdot w, y_i)$. Using the chain rule and evaluating at an arbitrary $\bar{w} \in \mathcal{V}$ we have

$$\begin{aligned}\partial_w[L(h(w, \cdot))](\bar{w}) &= \sum_i \partial_{\hat{y}}[\ell(\hat{y}_i, y_i)](J(x_i) \cdot \bar{w}) \cdot \partial_w(J(x_i) \cdot \bar{w}) \\ &= \sum_i \partial_{\hat{y}}[\ell(\hat{y}_i, y_i)](J(x_i) \cdot \bar{w}) \cdot J(x_i).\end{aligned}$$

Evaluating the affine function h consists of computing the Jacobian vector product $J(x_i)\bar{w}$. This can be done while avoiding computing the Jacobian explicitly by using forward mode automatic differentiation or finite differences. We find both approaches to work similarly well, with finite differences being slightly faster, and forward mode automatic differentiation more numerically stable. This chapter's experiments use finite differences, so we present this approach here. Specifically, we employ the method of Andrei (2009) to select the optimal step size. Chapter 6 will use automatic differentiation. We then evaluate the loss gradient at the linear model output, denoting this vector in our algorithm as $\mathfrak{g} = \partial_{\hat{y}}[\ell(\hat{y}, y)](J(x) \cdot \bar{w})$. This gradient can often be evaluated in closed form. Finally, we project $\mathfrak{g}$ onto the weights by multiplying with the Jacobian. This vector Jacobian product is implemented using automatic differentiation. That is, $\mathfrak{g}^T J(x_i) = \partial_v[\mathfrak{g}^T \cdot g(v, x_i)](\tilde{v})$. We combine these steps in algorithm 2.

Evaluating the gradient of $\|\bar{w}\|_A^2$ is trivial.

Algorithm 2: Efficient evaluation of the likelihood gradient for the linearised model

Inputs: Neural network g, Observation x, Linearisation point $\tilde{v}$, Weights to optimise w, Likelihood function $\ell(\cdot, y)$, Machine precision ϵ

1 $\delta = \sqrt{\epsilon}(1 + \|\tilde{v}\|_\infty)/\|w\|_\infty$ `// Set FD stepsize (Andrei, 2009)`

2 $\hat{y} = J(x) \cdot w \approx \frac{g(x,\tilde{v}+\delta w) - g(x,\tilde{v}-\delta w)}{2\delta}$ `// Two sided FD approximation to Jvp`

3 $\mathfrak{g} = \partial_{\hat{y}}[\ell(\hat{y}, y)](J(x) \cdot w)$ `// Evaluate gradient of loss at` $J(x) \cdot w$

4 $\mathfrak{g}^T \cdot J(x) = \partial_v[\mathfrak{g}^T \cdot g(v, x)](\tilde{v})$ `// Project gradient with backward mode AD`

Output: $\mathfrak{g}^T \cdot J(x)$

5.4.3 Further implications of our results

We now discuss details and implications of the presented recommendations and results.

Magnitude of linearisation point in normalised networks Optimising a normalised neural network returns a solution for the normalised weights (those in $\mathcal{V}''$) up to some scaling factor $\mathfrak{c} > 0$. How is c determined? Recall, from (5.9), that for any $v'' \in \mathcal{V}''$, the directional derivative of the NN output in the direction of v'' is zero. This is also illustrated in Figure 5.4. With this in mind, the dynamics of optimisation can be understood by analogy to a Newtonian system in polar coordinates. The weights are a mass upon which the data fit gradient acts as a tangential force. When discretised, this gradient pushes the weights away from zero. On the other hand, regularisation from the prior term acts like a centripetal force, pushing the weights towards the origin. The resulting c is thus proportional to the variance of the gradients of v'', and as such dependent on the learning rate and batch size hyperparameters, while being inversely proportional to the regularisation strength, e.g. weight decay. This has been studied extensively in the optimisation literature, including Cai et al. (2019); Hoffer et al. (2018); Li et al. (2020); Lobacheva et al. (2021); van Laarhoven (2017).

On network biases in the Jacobian feature expansion Most normalisation techniques introduce scale invariance by dividing subsets of network activations by an empirical estimate of their standard deviation. These activations depend on the values of both weights and biases. On the other hand, practical use of linearised Laplace commonly considers uncertainty due to only network weights (Daxberger et al., 2021b; Maddox et al., 2021), excluding bias entries from Jacobian and Hessian matrices. This departure from our assumptions can break the scale invariance necessary for lemma 6. Whether invariance is (approximately) preserved for the weights in the bias-exclusion setting depends on the relative effect of weights and biases on each subset of normalised activations. Invariance is preserved if the biases have small

impact. Empirically, we find that the inclusion (or exclusion) of biases does not alter the improvements obtained from applying our recommendations (see Figure 5.7).

Implications for the (non-linearised) Laplace method The (non-linearised) Laplace method (Kristiadi et al., 2020; Ritter et al., 2018) approximates the intractable posterior by means of a quadratic expansion around an optima, but without the linearisation step given in equation (5.2). As discussed in Section 5.2, when employing stochastic optimisation, early stopping, or normalisation layers, we will not find a minimiser of $\mathcal{L}_{g,A}$. Without a well-behaved surrogate linear model loss to fall back on, the Laplace method can yield very biased estimates of the model evidence.

5.5 Demonstration: hyperparameter selection with the tangent linear model

We proceed to provide empirical evidence for our assumptions and recommendations. Specifically, in Section 5.5.1, we validate the assumptions made in throughout this chapter. Then, in Section 5.5.2, we demonstrate that our recommendations yield improvements across a wide range of architectures. In these first two subsections, we employ networks containing at most 46k weights, since this is the largest model for which we can tractably compute the Hessian on an A100 GPU. This choice avoids confounding the effects described throughout the chapter with any further approximations. In Section 5.5.3, we show that our recommendations yield performance improvements on the 25M parameter ResNet-50 network while employing the KFAC approximation to the Hessian (Daxberger et al., 2021a; Martens and Grosse, 2015). Throughout, we focus on the layerwise prior precision, described in Section 5.3.1. We leave extensive evaluation of the g-prior for Chapter 6.

Unless specified otherwise, we: 1) train a NN to find $\tilde{v}$ using standard stochastic optimisation algorithms, 2) linearise the network about $\tilde{v}$ as in (5.2), 3) optimise the linear model weights using $\mathcal{L}_{h,A}$ (algorithm 2) and layer-wise regularisation parameters with $\mathcal{G}_{w_\star}$ (5.7), 4) compute the linearised predictive distribution with (3.39). We repeat this procedure with 5 random seeds and report mean results and standard error. For each seed, the methods compared produce the same mean predictions $g(\tilde{v}, \cdot)$, only differing in their predictive variance. In this setting, the test Negative Log-Likelihood (NLL, lower is better) can be understood as a measure of uncertainty miscalibration. The full set of experimental details for this chapter are provided in Chapter A.

5.5.1 Validation of modelling assumptions

We validate the key conjectures stated throughout the chapter. If not specified otherwise, we employ a 46k parameter ResNet (He et al., 2016a) with batch-normalisation after every convolutional layer. The output layer is dense, satisfying (5.10).

Choice of Hessian In Section 5.2, we suggest evaluating the Hessian of $\mathcal{L}_h$ at the linearisation point $\tilde{v}$ (instead of $w_\star$) for model evidence optimisation (5.7). This avoids the need to recompute the Hessian throughout optimisation. Figure 5.7 (left) shows how the improvement from using the recommended model evidence $\mathcal{G}_{w_\star}$, as opposed to $\mathcal{G}_{\tilde{v}}$, dominates the effect of the choice of Hessian evaluation point.

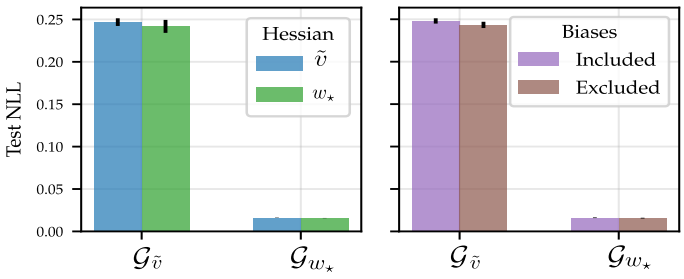


Fig. 5.7 Comparison of the test NLL improvement obtained when switching from $\mathcal{G}_{\tilde{v}}$ to $\mathcal{G}_{w_\star}$ to optimise the prior precision A relative to the impact of (*left*) evaluating the Hessian at $\tilde{v}$ or $w_\star$, and (*right*) excluding network biases from the basis functions. Both plots use a $d = 46k$ ResNet with batch norm trained on MNIST.

Dependence on $\mathfrak{c}$ for isotropic precisions In Figure 5.5, we illustrate the dependence of the predictive posterior on the scale of normalised weights $\mathfrak{c}$ for an isotropic prior precision $A = aI$, i.e. recommendation 2 is ignored. We use a 2.6k parameter 2 hidden layer fully connected NN with layer norm after every layer except the last and a 1d regression task. Changing $\mathfrak{c}$ changes the optimal a and, consequently, the predictive uncertainty changes. With layer-wise a this effect vanishes (as predicted by proposition 5).

Treatment of NN biases Excluding the Jacobians of network biases from our basis function expansion breaks the scaling properties presented in lemma 6. In Figure 5.7 (right), we show that the effect of excluding biases is dominated by the choice of the model evidence between $\mathcal{G}_{w_\star}$ and $\mathcal{G}_{\tilde{v}}$.

Early stopping We evaluate whether more thorough optimisation of the NN weights with $\mathcal{L}_g$ leads to a linearisation point $\tilde{v}$ closer to $w_\star$ in the sense of the implied optimal regularisation and induced posterior predictive distribution. We perform this analysis on normalised and

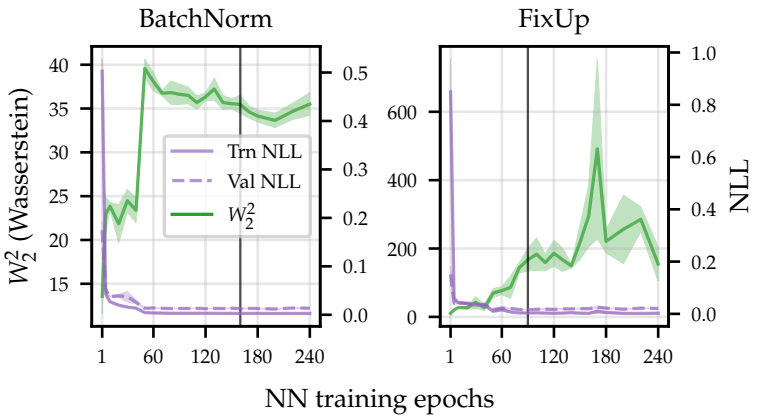


Fig. 5.8 Wasserstein distance between predictive posteriors obtained when using $\mathcal{G}_{\tilde{v}}$ and $\mathcal{G}_{w_\star}$ throughout NN training (i.e. the linearisation point $\tilde{v}$ is changing). The vertical black line indicates optimal (val-based) early stopping.

unnormalised networks (for which use the non scale-invariant FixUp regularisation instead (Zhang et al., 2019)), since $\tilde{v}$ is guaranteed to never match $w_\star$ for the former. Surprisingly, the Wasserstein-2 distance between predictive distributions obtained with $\mathcal{G}_{\tilde{v}}$ and $\mathcal{G}_{w_\star}$ increases with more optimisation steps in both cases. Thus, more thorough optimisation does not help.

		Transformer	CNN	ResNet	Pre-ResNet	FixUp	U-Net
$\mathcal{G}_{w_\star}$	single a	**0.162** ± 0.042	**0.025** ± 0.000	**0.017** ± 0.000	0.017 ± 0.000	**0.055** ± 0.006	-1.793 ± 0.050
	layerwise a	**0.162** ± 0.042	**0.025** ± 0.000	**0.016** ± 0.001	**0.016** ± 0.000	0.061 ± 0.005	**-2.240** ± 0.027
$\mathcal{G}_{\tilde{v}}$	single a	0.310 ± 0.060	0.253 ± 0.001	0.252 ± 0.006	0.220 ± 0.004	0.153 ± 0.021	-1.164 ± 0.052
	layerwise a	**0.162** ± 0.042	0.205 ± 0.002	0.236 ± 0.005	0.239 ± 0.004	0.200 ± 0.018	-1.703 ± 0.023

Table 5.1 Validation of recommendations across architectures. All results are reported as negative log-likelihoods (lower is better). In each column, the best performing method is bolded. For each $\mathcal{M}$, if single or layerwise a optimisation performs better, it is underlined.

5.5.2 Validating recommendations across architectures

We evaluate the utility of recommendation 1, and recommendation 2 on a range of architectures and tasks: 1) a transformer architecture on the pointcloud-MNIST variable length sequence classification task. This model uses layer norm in alternating layers, 2) a LeNet-style CNN with batch norm placed after every convolutional layer and a dense output layer tasked with MNIST classification, 3) a ResNet with batch norm after every layer except the dense output layer (MNIST classification), 4) the same ResNet but with batch norm substituted by (non scale-invariant) FixUp regularisation (MNIST classification). 5. a pre-ResNet (He et al.,

2016b). This architecture differs from ResNet in that batch norm is placed before each weight layer instead of after them; the implication is that there is only 1 normalised group of weights encompassing all weights but those of the dense output layer (MNIST classification). 6. a fully convolutional U-net autoencoder tasked with tomographic reconstruction (regression) of a KMNIST character from a noisy low-dimensional observation. We reproduce the experimental setting of Barbano et al. (2022c) for this task. Group norm is placed after every layer except the last, which is convolutional.

As shown in Table 5.1, the application of recommendation 1 yields notably improved performance across all settings. Applying recommendation 2 yields modest improvements for classification networks with normalisation layers but large improvements for the U-net. Interestingly, layer-wise regularisation degrades performance in the (non-normalised) FixUp ResNet.

5.5.3 Large scale models

		BATCH NORM	FIXUP
$\mathcal{G}_{w_\star,\text{simple}}$	single a	**0.112** $\pm$ 0.004	0.128 $\pm$ 0.000
	layerwise a	**0.109** $\pm$ 0.003	**<u>0.096</u>** $\pm$ 0.000
$\mathcal{G}_{w_\star}$	single a	0.190 $\pm$ 0.005	0.249 $\pm$ 0.002
	layerwise a	0.194 $\pm$ 0.009	<u>0.193</u> $\pm$ 0.001
$\mathcal{G}_{\tilde{v}}$	single a	0.570 $\pm$ 0.004	0.412 $\pm$ 0.000
	layerwise a	0.567 $\pm$ 0.004	<u>0.360</u> $\pm$ 0.000

Table 5.2 Test negative log-likelihoods for ResNet-50 on CIFAR10.

We validate our recommendations on the 25M parameter ResNet-50 network trained on the CIFAR10 dataset. This model places batch norm after every layer except the dense output layer. We also consider a normalisation-free FixUp ResNet-50. Table 5.2 shows that both of our recommendations yield better test NLL, with larger gains obtained by the batch norm network. The normalisation-free FixUp setting does not simplify as in eq. (5.10). Nonetheless, assuming the simplified model evidence, denoted $\mathcal{G}_{v_\star,\text{simple}}$, when obtaining the linear model optima yields improved performance for all models.

5.6 Discussion

This chapter has identified and addressed two pitfalls of a naïve application of linearised Laplace to modern NNs in the *post-hoc* setting. First, the optima of the loss function is not found in practice. This invalidates the assumption that the point at which we linearise our model is stationary. However, every linearisation point implies an associated basis function linear model. As we use this model to provide errobars, we propose to choose hyperparameters using the evidence of this model. This requires only the solving of a convex optimisation problem, one much simpler than NN optimisation. Second, normalisation layers introduce an invariance to the scale of NN weights and thus the linearisation point can only be identified up to a scaling factor. We show that to obtain a predictive posterior that is invariant to this scaling factor, the regulariser must be independently parametrised for each normalised group of weights, e.g. different layers. We also show that a classical feature normalisation method, the g-prior, solves this issue. Our experiments confirm the effectiveness of these recommendations across a wide range of model architectures and sizes.

With these advancements, and the scalable SGD-based sampling from Chapter 4, we are almost ready to perform Bayesian inference and hyperparameter optimisation with large scale linearised neural networks. The only remaining impediment is computing the log-determinant term in the expression for the model evidence. Chapter 6 will provide the final piece of the puzzle by introducing an accurate method to learn the linearised Laplace prior precision using only posterior samples.

Chapter 6

Scalable uncertainty estimation and hyperparameter learning for neural networks with sample-based linearised Laplace inference

"One thing that should be learned (...) is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great." — Richard Sutton

The linearised Laplace method, originally introduced by Mackay (1992a), and reviewed in Section 3.3, has received renewed interest in the context of uncertainty quantification for modern neural networks (NN) (Daxberger et al., 2021a; Immer et al., 2021b; Khan et al., 2019b). The method constructs a surrogate Gaussian linear model for the NN predictions, and uses the error bars of that linear model as estimates of the NN's uncertainty. However, the resulting linear model is very large; the design matrix is sized number of parameters by number of datapoints times number of output classes. Thus, both the primal (weight space) and dual (observation space) formulations of the linear model are intractable. This restricts the method to small network or small data settings. Moreover, the method is sensitive to the choice of regularisation strength for the linear model (Antorán et al., 2022; Immer et al., 2021a). This chapter develops methods to scale inference and hyperparameter selection to very large linear models with a particular focus on linearised neural networks.

To scale inference and hyperparameter selection in Gaussian linear regression, we introduce a sample-based Expectation Maximisation (EM) algorithm. It interleaves E-steps, where we infer the model's posterior distribution over parameters, given some choice of hyperparameters, and M-steps, where the hyperparameters are improved given the current posterior. Our contributions here are two-fold:

1. We perform posterior sampling for large-scale linearised neural networks using stochastic gradient descent with the low-variance sample-then-optimise objective introduced in Section 4.2.2, which we use to approximate the E-step.

2. We introduce a method for hyperparameter selection that only requires access to posterior samples, and not the full posterior distribution. This forms our M-step.

Combined, these allow us to perform inference and hyperparameter selection by solving a series of quadratic optimisation problems using stochastic gradient descent, and thus avoiding an explicit cubic cost in any of the problem's properties. Our method readily extends to non-conjugate settings, such as classification problems, through the use of the Laplace approximation. In the context of linearised NNs, our approach also differs from previous work in that it avoids instantiating the full NN Jacobian matrix, an operation requiring as many backward passes as output dimensions in the network.

We demonstrate the strength of our inference technique in the context of the linearised Laplace procedure for image classification on CIFAR100 (100 classes × 50k datapoints) and Imagenet (1000 classes × 1.2M datapoints) using an 11M parameter ResNet-18 and a 25M parameter ResNet-50, respectively. The methods introduced in this chapter will allow us to perform uncertainty estimation in high-resolution volumetric tomographic image reconstruction in Chapter 7.

The rest of this chapter is organised as follows. Section 6.1 introduces a variational EM algorithm for linearised neural networks. Section 6.2 discusses a series of methods to scale up the aforementioned algorithm to the large model and large dataset setting. Section 6.3 demonstrates these methods on large-scale image classification. Finally, Section 6.4 concludes the chapter.

6.1 Variational EM for linearised neural networks

We consider the multioutput conjugate Gaussian linear model class, introduced in Chapter 2, and which we review here. Our choice of basis functions are induced by a first order Taylor

expansion of a NN $g : \mathcal{V} \times \mathcal{X} \to \mathbb{R}^c$ around its pre-trained parameters $\tilde{v} \in \mathcal{V} \subseteq \mathbb{R}^d$. We will work with fully normalised networks with a dense final layer[1]. In Section 5.4.1 we showed that, when linearised, these take the simplified form

$$h(w, \cdot) = \phi(\cdot)w,$$

where $\phi(x) = \partial_v g(\tilde{v}, x) \in \mathbb{R}^{c \times d}$ is the NN's Jacobian evaluated at $x \in \mathcal{X}$, which acts as a feature expansion of the input.

With that, the generative process we assume relates our inputs $x_1, \ldots, x_n \in \mathcal{X}$ and corresponding outputs $y_1, \ldots, y_n \in \mathcal{Y} \subseteq \mathbb{R}^c$ is

$$Y = \Phi w + \mathcal{E} \quad \text{with} \quad w \sim \mathcal{N}(0, A^{-1}I) \quad \text{and} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}),$$

where $Y \in \mathbb{R}^{nc}$ is the concatenation of $y_1, \ldots, y_n$, B is a block diagonal matrix, built from $(B_i)_{i=1}^n$, a set of $c \times c$ blocks representing the noise precision for each iid observation, and $\Phi = [\phi(x_1)^T; \ldots; \phi(x_n)^T]^T \in \mathbb{R}^{nc \times d}$ is the embedded design matrix. We define $M = \Phi^T B \Phi \in \mathbb{R}^{d \times d}$, which matches the curvature of the Gaussian likelihood. Finally, $A \in \mathbb{R}^{d \times d}$ is a positive definite prior precision matrix, which we treat as a hyperparameter.

6.1.1 Conjugate Gaussian regression and the EM algorithm

Our goal is to infer the posterior distribution for the parameters w given our observations, under the setting of A most likely to have generated the observed data. We use an iterative procedure inspired by Mackay (1992a), which alternates computing the posterior for w, denoted $\Pi_{w|Y}$, for a given choice of A, and updating A, until the pair $(A, \Pi_{w|Y})$ converge to a locally optimal setting. This corresponds to an EM algorithm (Bishop, 2006; Dempster et al., 1977).

With that, we start with some initial $A \in \mathbb{R}^{d \times d}$, and iterate:

- (E step) Given A, the posterior for w, denoted $\Pi_{w|Y}$, is computed exactly as

$$\Pi_{w|Y} = \mathcal{N}(w_\star, H^{-1}) \quad \text{where} \quad H = M + A \quad \text{and} \quad w_\star = H^{-1}\Phi^T B Y. \tag{6.1}$$

[1] In Chapter 7, we will work with models without a dense final layer. Fortunately, the procedures discussed in this chapter may be applied out-of-the-box to this setting by shifting the linear model targets by the constant-in-w terms in the NN's Taylor expansion.

- (M step) We lower bound the log-probability density of the observed data, i.e. the evidence, for the model with posterior $\Pi_{w|Y}$ and precision A' as

$$\log p(Y; A') \geq -\tfrac{1}{2}\|w_\star\|^2_{A'} - \tfrac{1}{2}\log \det(I + A'^{-1}M) + C =: \mathcal{M}(w_\star, A'), \qquad (6.2)$$

for C independent of A'. We choose a new setting for A that improves this lower bound.

Derivation Derivation of (6.2) as a lower bound on the evidence

To show this we part from the Gaussian ELBO for the Gaussian-linear model given in (3.5)

$$\begin{aligned}\log p(Y; A) \geq \mathcal{M}(w_q, \Sigma_q, A) = \frac{1}{2}\big(&- n\log(2\pi) - \log \det B^{-1} - \log \det A^{-1} \\ &- \|w_q\|^2_A - \|Y - \Phi w_q\|^2_B + \log \det \Sigma_q \\ &- \mathrm{Tr}(\Phi\Sigma_q\Phi^T B) + d - \mathrm{Tr}(\Sigma_q A)\big).\end{aligned}$$

and choose $\Sigma_q = (M + A)^{-1} = H^{-1}$, which is the optimal setting, for any value of w_q and A. With this we note that

$$\mathrm{Tr}\left(\Phi H^{-1}\Phi^T B\right) = d - \mathrm{Tr}(H^{-1}A) = \gamma,$$

are both expressions for the effective dimension (see (2.46)), which cancel out. This leaves us with

$$\begin{aligned}\mathcal{M}(w_q, A) = \frac{1}{2}\big(&- \|w_q\|^2_A - \log \det A^{-1} + \log \det \Sigma_q \\ &- \|Y - \Phi w_q\|^2_B - n\log(2\pi) - \log \det B^{-1}\big),\end{aligned}$$

which matches (6.2) when we set w_q to $w_\star$ and identify the constant in A terms as $C = \frac{1}{2}(-\|Y - \Phi w_q\|^2_B - n\log(2\pi) - \log \det B^{-1}) = \log p(Y|w_q; B)$.

Remark On the ELBO in (6.2)

$\mathcal{M}(w_\star, A)$ has a variational parameter, the posterior mean and a hyperparameter the prior precision. For each prior precision, there is an optima posterior mean which makes the bound tight. At each M step, we update our prior precision, and the variational

posterior's covariance updates automatically with the new regulariser, leaving the posterior mean as the only variational parameter to be found anew in successive E steps. Because the log-likelihood is quadratic, its curvature M is fixed throughout the EM iteration.

6.1.2 Laplace-approximating non-conjugate likelihoods

We now consider the setting where the linearised model's loss, defined as

$$\mathcal{L}_{h,A}(v) = L(h(w, \cdot)) + \|w\|_A^2, \tag{6.3}$$

for $L\colon \mathcal{Y}^{\mathcal{V}\times\mathcal{X}} \mapsto \mathbb{R}_+$ of the form $L(h(w,\cdot)) = \sum_i^n \ell(y_i, h(w, x_i))$, and ℓ is a negative log-likelihood function, is non quadratic. That is, L corresponds to a *non-Gaussian* density.

We employ the Laplace approximation (see Section 3.3.2 for a review) for the E-step. That is, we construct a Gaussian approximate posterior as

$$\mathcal{N}(w_\star, H^{-1}) \quad \text{with} \quad w_\star = \arg\min_{w\in\mathbb{R}^d} \mathcal{L}_{h,A}(v)$$
$$\text{and} \quad H = \Phi^T B \Phi + A \quad .$$

Here, $B \in \mathbb{R}^{nc\times nc}$ is a again a block diagonal matrix built from blocks $B_i = \partial^2_{\hat{y}_i} \ell(y_i, \hat{y}_i)$ which we evaluate at predictions $\hat{y}_i = h(\tilde{v}, x_i) = g(\tilde{v}, x_i)$ in place of $h(w_\star, x_i)$, since the latter would change each time the regulariser A is updated, requiring expensive re-evaluation. This decision was recommended in Section 5.2 and ablated in Section 5.5.1.

We plug in the above expressions into the Laplace evidence, given in (6.2), for the M step. However, this may no longer represent a lower bound on the true evidence. The EM procedure from Section 6.1.1 is for the conjugate Gaussian-linear model, where it carries guarantees on non-decreasing model evidence, and thus convergence to a local optimum. These guarantees do not hold for non-conjugate likelihood functions, e.g., the softmax-categorical, where the Laplace approximation is necessary. Instead, we are guaranteed convergence to a local optima of the evidence of a surrogate model with Laplace approximated likelihood.

6.1.3 The issue of limited scalability

The above inference and hyperparameter selection procedure for $\Pi_{w|Y}$ and A is futile when both d and nc are large. The E-step requires the inversion of a $d \times d$ matrix and the M-step evaluating its log-determinant, both cubic operations in d. These may be rewritten to instead yield a cubic dependence on nc (as in Section 2.2.1), but under our assumptions, that too is not computationally tractable. Instead, we now pursue a stochastic approximation to this EM-procedure.

6.2 Sample-based inference for the tangent linear model

We now present the chapter's main contribution, a stochastic approximation (Nielsen, 2000) to the iterative algorithm presented in the previous section. Our M-step, presented in Section 6.2.1, requires only access to samples from $\Pi_{w|Y}$. We then touch on a number of practical and implementation matters. We provide an efficient implementation of the g-prior in Section 6.2.2. Section 6.2.3 discusses an efficient implementation of the SGD posterior sampling methods introduced in Chapter 4. These constitute our E-step. We discuss efficient sample-based predictions for linearised neural networks in Section 6.2.4. We conclude with a full description of our inference algorithm in Section 6.2.5, with special attention to its application to image classification.

6.2.1 Hyperparameter learning using posterior samples

For now, assume that we have an efficient method of obtaining samples from a zero-mean version of the posterior $\zeta_1, \ldots, \zeta_k \sim \mathcal{N}(0,\, H^{-1}) \coloneqq \Pi^0_{w|Y}$, and access to $w_\star$, the mean of $\Pi_{w|Y}$. Evaluating the first order optimality condition for $\mathcal{M}(w_\star, A)$ yields that the optimal choice of A satisfies

$$\|w_\star\|_A^2 = \mathrm{Tr}\{H^{-1}M\} =: \gamma, \tag{6.4}$$

where the quantity γ is the effective dimension of the regression problem (see Section 2.4.4). It can be interpreted as the number of directions in which the weights w are strongly determined by the data. Setting $A = aI$ [2] for $a = \gamma/\|w_\star\|^2$ yields a contraction step converging towards the optimum of $\mathcal{M}$ (Mackay, 1992a). We thus hereon refer to such a contraction step as a *MacKay update*.

[2] We absorb additional prior structure into the basis functions in Section 6.2.2

Derivation First order optimality condition
Consider the derivative of $\mathcal{M}$. We have,

$$\partial_A \log p(Y; A) = -\frac{1}{2}\left[\partial_A \|w_\star\|_A^2 + \partial_A \log \det(A + M) - \partial_A \log \det A\right], \tag{6.5}$$

where we expanded $\log \det(I + A^{-1}M) = \log \det(A + M) - \log \det A$. Taking the respective derivatives and setting equal to zero at A, this leads to the condition

$$w_\star w_\star^T = (I - (I + A^{-1}M)^{-1})A^{-1}. \tag{6.6}$$

Post-multiplying by A and applying the push-through identity, we obtain

$$w_\star w_\star^T A = M(A + M)^{-1}. \tag{6.7}$$

For the above to hold, it is necessary that the traces of both sides are equal. Thus,

$$\|\tilde{w}\|_A^2 = \text{Tr}\{\tilde{w}\bar{w}^T A\} = \text{Tr}\{M(A + M)^{-1}\} = \gamma, \tag{6.8}$$

which is the stated first order optimality condition, up to a cyclic permutation.

Computing γ directly requires the inversion of H, a cubic operation. We instead rewrite γ as an expectation with respect to $\Pi^0_{w|Y}$ using Hutchinson (1990)'s trick, and approximate it using samples as

$$\begin{aligned}\gamma = \text{Tr}\{H^{-1}M\} = \text{Tr}\{H^{-\frac{1}{2}}MH^{-\frac{1}{2}}\} &= \mathbb{E}_{\zeta_1 \sim \Pi^0_{w|Y}}[\zeta_1^T M \zeta_1] \\ &\approx \tfrac{1}{k}\textstyle\sum_{j=1}^k \zeta_j^T \Phi^T B \Phi \zeta_j := \hat{\gamma}.\end{aligned} \tag{6.9}$$

We then select $a = \hat{\gamma}/\|w_\star\|^2$. We have thus avoided the explicit cubic cost of computing the log-determinant in the expression for $\mathcal{M}$ (given in (6.2)) or inverting H. Due to the block diagonal structure of B, $\hat{\gamma}$ may be computed in $\mathcal{O}(n)$ Jacobian vector products as $\hat{\gamma} = \frac{1}{k}\sum_{j=1}^k \sum_{i=1}^n \zeta_j^T \phi(x_i)^T B_i \phi(x_i)\zeta_j$.

Demonstration: MacKay's effective-dimension-based M-step

We empirically motivate the fixed-point iteration M-step introduced by Mackay (1992a), by comparing it with alternative approaches to updating hyperparameters. In particular, we compare MacKay's update with the standard Laplace M-step evidence, denoted $\mathcal{M}$ and given

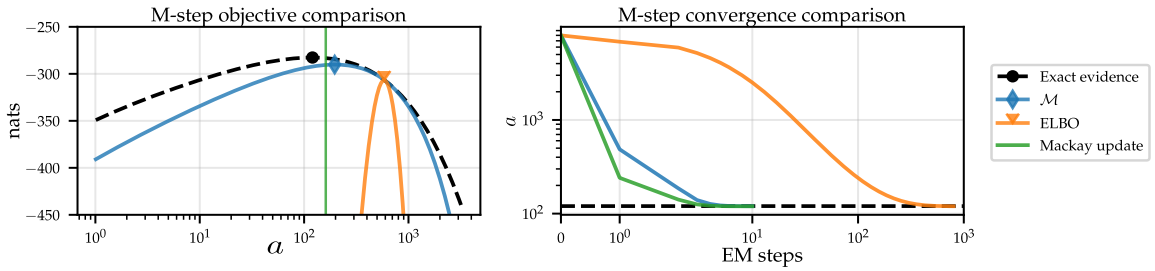


Fig. 6.1 Left: exact model evidence for a linearised 2 hidden layer MLP with layer normalisation together with the lower bound presented in (6.2), $\mathcal{M}$, and an ELBO where the Gaussian posterior covariance is decoupled from the regulariser. All curves use an initial regulariser of $a = 500$ and have a marker placed at their optima. The value proposed by the MacKay update is marked with a vertical green line. Right: values of the regularisation strength a obtained at successive EM iterations while using the different update strategies under consideration for the M step. Note that when we assume access to the exact evidence function, the regulariser converges in a single step and no EM iteration is necessary.

in (6.2), and a Gaussian ELBO, of the form given in (3.5), with both the mean and covariance acting as variational parameters. The latter two approaches differ in that the ELBO's posterior covariance is not clamped to the optimum value for the current regulariser A, and thus the bound is less tight. That is, ELBO's covariance does not change with the regulariser while performing the M-step. Both of these objectives differ from the MacKay update in that they provide an objective which requires gradient-based optimisation in the M-step. Instead, the MacKay update has a closed-form.

The plot on the left of Figure 6.1 compares the exact linearised Laplace evidence for a 2 hidden layer MLP with layernorm trained on the toy dataset of Antorán et al. (2020) with the bound $\mathcal{M}$ (6.2) and with the decoupled ELBO (6.2). We evaluate all of these exactly, without resorting to Monte Carlo sampling. The initial regulariser is set to $a = 500$. The ELBO is only tight for regulariser values very close to initialisation, resulting in very small M steps. $\mathcal{M}$ is tangent to the evidence at the same point as the ELBO but presents a much better approximation as we move away from $a = 500$. The optimum of $\mathcal{M}$ is much closer to the optimum of the evidence. The MacKay update does not use a lower bound but instead provides an updated value for a which is even closer to the optimum of the evidence. The right hand side plot shows the change in the regularisation parameter across successive M-steps using the update methods under consideration. The MacKay M-step converges to the optima of the evidence in 2 steps. Using $\mathcal{M}$ as an objective results in convergence after 5 steps. On the other hand, the ELBO update requires around 100 steps. Figure 6.2 further illustrates hyperparameter learning in the 1d toy setting by showing the successive lower

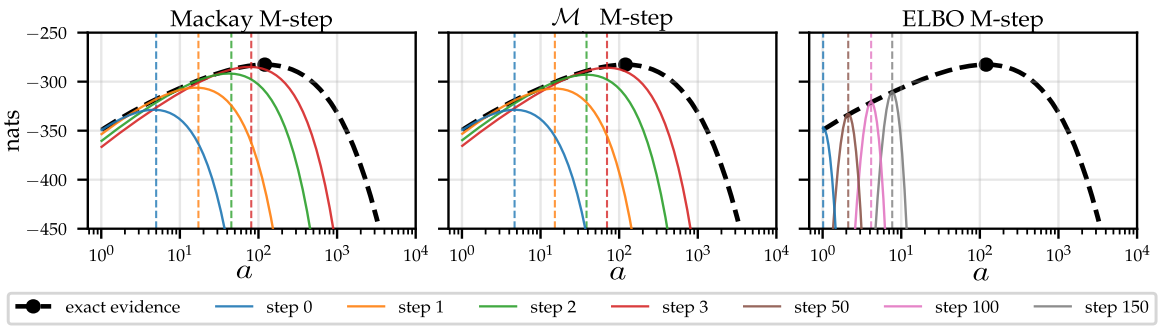


Fig. 6.2 Exact linear model evidence for a linearised 2 hidden layer MLP with layer normalisation together with the lower bound presented in (6.2), $\mathcal{M}$ (left and middle plots), and an ELBO, where the Gaussian posterior covariance is decoupled from the regulariser (right side plot), at different EM steps. We update the regularisation strength with MacKay's fixed point iteration for the left side plot. Note that $\mathcal{M}$ curves are shown in this plot. We maximise $\mathcal{M}$ in the middle plot and we maximise the ELBO in the right hand side plot. All curves use an initial regulariser of $a = 5$ and we place a vertical dashed line at each step's update. Starting below the optimal regularisation strength makes convergence behaviour differ from that of Figure 6.1, which starts from above the optima.

bounds obtained by each of the approaches under consideration at each M-step. Interestingly, the MacKay update produces regulariser updates that almost exactly maximise $\mathcal{M}$.

Demonstration: Comparing estimators of the effective dimension

The effective dimension estimator introduced in (6.9) is in kernelised form. A different unbiased estimator may be obtained in weight-space form following the derivation provided in (2.46). That is

$$d - \text{Tr}\left(AH^{-1}\right) \approx d - \frac{1}{k}\sum_{j=1}^{k} \zeta_j^T A \zeta_j \quad \text{with} \quad \zeta_j \sim \Pi^0_{w|Y} \tag{6.10}$$

Figure 6.3 compares both estimators when applied to the 1d toy problem used to generate Figure 6.5 from the main text. In particular, we use a linearised 2 hidden layer MLP with 50 hidden units and layernorm after every hidden layer ($d = 2700$). We use the "Matérn" dataset of Antorán et al. (2020). We use 8 samples from the exact linearised Laplace posterior to compute effective dimension estimates and repeat this procedure 1000 times to characterise the behaviour of each estimator. As a reference, we also compute the exact effective dimension using eigendecomposition.

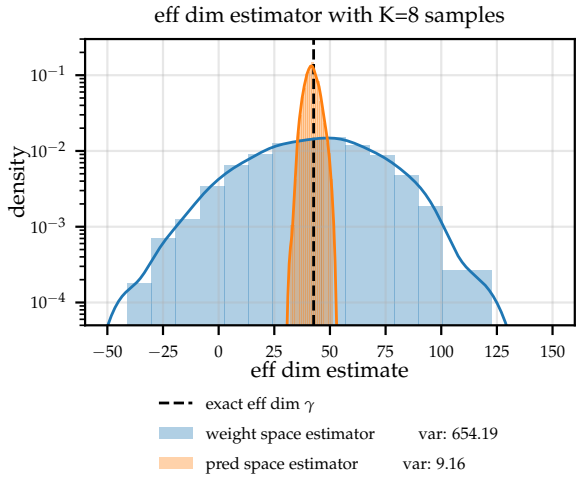


Fig. 6.3 Histogram, with bin heights normalised to represent density estimates, of the effective dimension estimates produced by the primal form (weight space) estimator (6.10) and the kernelised (prediction space) estimator (6.9). Both distributions are roughly centred at the true effective dimension but the kernelised estimator presents much lower variance.

Both estimators present distributions centred at the true effective dimension value. However, the prediction space (kernelised) estimator presents a much lower variance of 9.16 as opposed to 654.19 for the weight space estimator. Additionally, the weight space estimator distribution places a substantial amount of probability mass on negative effective dimension values. From the form of (6.10), we see that this is due to our 8-sample estimator overestimating posterior variance. On the other hand, the kernelised estimator in (6.9) can only produce positive values.

Remark Extension of the MacKay update to layerwise prior precision parameters

We can leverage the primal form expression for the effective dimension given in (6.10) to extend the MacKay update to the layer-wise regulariser setting (see Section 5.3.1). Consider a sub-vector of our weight vector contiguous between the ith and jth weights written as $w_{\star i:j}$. Note that we only choose contiguous weights for notational convenience but it is not necessary to do so in general.
The first order optimality condition is satisfied if for any i, j with $i < j$, we have

$$\sum_{k=i}^{j}[A]_{kk}w_{\star_k}^2 = j - i - \sum_{k=i}^{j}[A]_{kk}[(A+M)^{-1}]_{kk} \coloneqq \gamma_{i:j}. \tag{6.11}$$

We assume $[A]_{kk} = a$ for all $i \leq k < j$. Thus, we may update the regulariser for each separate weight sub-vector as $a = \gamma_{i:j}/\|w_{\star i:j}\|^2$. However, we find this leads to slower convergence than when estimating a single prior precision for the whole model. Combined with the weight-space estimator of the effective dimension (6.10) presenting higher variance, this make layerwise prior precision estimation with the MacKay update less attractive.

6.2.2 Constructing an efficient estimator of the g-prior

We use the diagonal g-prior, introduced in Section 5.3.2, with with a prior precision of the form $a \operatorname{diag} M$, where the diag operator takes as input a matrix and returns a diagonal version of the matrix. This leaves a single free parameter $a \in \mathbb{R}_+$, which will be estimated using MacKay updates, as described in Section 6.2.1. However, this requires the prior precision to be isotropic. We achieve this by absorbing the scaling structure $\operatorname{diag} M$ into our feature expansion as

$$\phi'(x) = \phi(x) \operatorname{diag}(s) \quad \text{for} \quad s \in \mathbb{R}^d \quad \text{with entries} \quad s_i = [M]_{ii}^{-1/2}, \tag{6.12}$$

where $i \leq d$ and $\operatorname{diag} s$ denotes a diagonal matrix with entries given by the vector s. And thus we work with the scaled Jacobian features $\phi'(\cdot)$ throughout, while assuming a prior precision of the form $A = aI$. Notice that the covariance kernels implied by these expansions match $a\phi'(\cdot)\phi'(\cdot')^T = a\phi(\cdot)\operatorname{diag}(M)\phi(\cdot')^T$; our generative model is unchanged.

We now turn to computing the scaling vector s. Naïvely, each entry would be computed as $s_j = \left(\sum_{i=1}^n e_j^T \phi(x_i)^T B_i \phi(x_i) e_j\right)^{\frac{1}{2}}$ for e_j $j \leq d$ the unit vectors corresponding to the canonical basis for the euclidean space $\mathbb{R}^d$. This would require $\mathcal{O}(nd)$ Jacobian vector products, which is intractable for large models and datasets.

Instead we stochastically estimate s using k samples as

$$s = \mathbb{E}_{\mathcal{E}\sim\mathcal{N}(0,B^{-1})}(\Phi^T B\mathcal{E})^{\odot -0.5} \approx \left(\frac{1}{k}\sum_{j=1}^{k}(\Phi^T B\mathcal{E}_j)^{\odot 2}\right)^{\odot -0.5} \quad \text{with} \quad \mathcal{E}_j \sim \mathcal{N}(0, B^{-1}), \tag{6.13}$$

where $^{\odot}$ refers to the elementwise power. The number of Jacobian vector products needed is now $\mathcal{O}(nk)$.

6.2.3 Efficient SGD posterior sampling with warm starts

All that is left is obtaining the linear model's posterior mean $w_\star$ and sampling from the 0 mean posterior $\Pi^0_{w|Y}$ for the E-step. For the former, we target the liner model's loss function $\mathcal{L}_{h,A}$, given in (6.3), with stochastic gradient descent. For the latter, we use the low variance weight-space sampling objective introduced in Section 4.2.2 and which we re-state here for the reader's convenience

$$\frac{1}{2}\|\Phi w\|_B^2 + \frac{1}{2}\|w - w_0'\|_A^2 \quad \text{with} \quad w_0' = w_0 + A^{-1}\Phi^T B\mathcal{E} \tag{6.14}$$
$$\text{where} \quad \mathcal{E} \sim \mathcal{N}(0, B^{-1}) \quad \text{and} \quad w_0 \sim \mathcal{N}(0, A^{-1}).$$

Notice how we can re-use samples used to estimate the g-prior scaling vectors $\Phi^T B\mathcal{E}_j$ in (6.13) to compute the regulariser target w_0'.

In order to limit computational cost, we sample the stochastic regularisation terms w_0', only once, and keep them fixed throughout EM iteration. This results in the optima of the sampling objective being close for successive iterations with different regularisation strength values. This comes at the cost of a small bias in our estimator which we find to be negligible in practise. We separate w_0' into a sum consisting of a prior sample from w_0 and a data dependent term, denoted $A^{-1}\Phi^T B\mathcal{E}$. The former scales with $a^{-1/2}$ while the latter with a^{-1}. This allows us to update each term in closed form each time a changes in the M step. We initialise our posterior samples at w_0 at the first EM iteration and warm start them with the previously optimised values in successive iterations. Similarly, we warm-start the posterior mode $w_\star$ at the previous solution between iterations, initialising it to zero for the first iteration. We optimise both our samples and posterior mean using stochastic gradient descent with Nesterov momentum. In particular, we follow the recommendations given in Section 4.2.4. We only depart from this for non-quadratic likelihoods, like softmax cross entropy, where we substitute geometric iterate averaging with a linearly decreasing step-size schedule (Bach, 2014). As a preview of this procedure, we display the SGD optimisation traces for the posterior mean $w_\star$ and samples ζ throughout all steps of our EM procedure for a linearised ResNet-18 trained on the CIFAR100 dataset in Figure 6.4.

Remark The impact of SGD's bias on our hyperparameter updates
Letting $B = bI$ for simplicity, our estimate of the effective dimension amounts to estimating the K^2 norm, or output space norm of our samples $\mathbb{E}_\zeta b\|\Phi\zeta\|^2 = b\|h(\zeta, X)\|^2$ (6.9). The other term appearing in the MacKay update is the A norm of our posterior

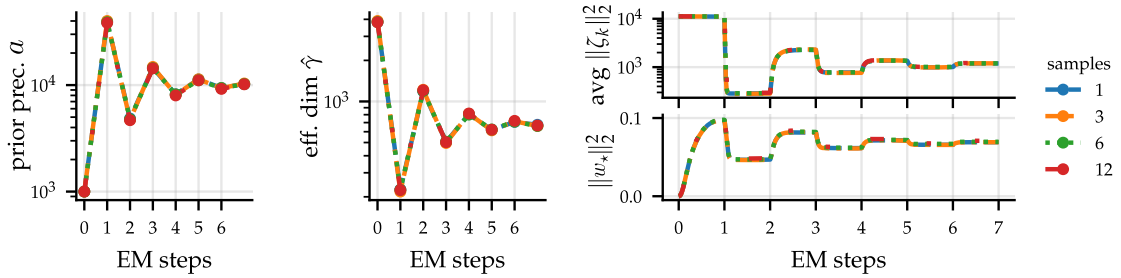


Fig. 6.4 Left: prior precision optimisation traces for ResNet-18 on CIFAR100 varying n. samples. Middle: same for the eff. dim. Right: average sample norm and posterior mean norm throughout successive EM steps' SGD runs while varying n. samples. Note that traces almost perfectly overlap. 1 posterior sample is enough to obtain a very accurate estimate of the effective dimension. As a result, the optimisation traces corresponding to different numbers of samples almost perfectly overlap.

mean $\|w_\star\|_A^2$. In Chapter 4 we saw how SGD converges quickly in the output space, but slowly in the weight space, both in an L_2 sense (see Figure 4.9). As a result, we expect to obtain an accurate estimate of the effective dimension, but not of $\|w_\star\|_A^2$. Given that we initialise the MAP setting of the weights at 0 for SGD optimisation, we expect that SGD will result is us underestimating $\|w_\star\|_A^2$. In turn, this will lead to overestimation of our regulariser when setting it with $a = \gamma / \|w_\star\|_A^2$. Indeed, this issue will appear in very large scale problems in Section 6.3 and in Section 7.4.2.

6.2.4 Sample-based linearised Laplace predictions

The linearised Laplace distribution over function outputs at an input x is the Gaussian $\mathcal{N}(g(\tilde{v}, x), \phi(x) H^{-1} \phi(x)^T)$. Here, we are following Section 3.3 and Chapter 5 in using the neural network output $g(\tilde{v}, \cdot)$ as the predictive mean, rather than the surrogate model mean $h(w_\star, \cdot)$. However, even given H^{-1}, evaluating this naïvely requires instantiating $\phi(x)$, at a cost of c vector-Jacobian products (i.e. backward passes). This is prohibitive for large c. However, expectations of any function $\sigma : \mathbb{R}^c \to \mathbb{R}$ under the predictive posterior can be approximated using only samples from $\Pi^0_{w|Y}$ as

$$\mathbb{E}_{\Pi_{w|Y}}[\sigma] \approx \frac{1}{k} \sum_{j=1}^{k} \sigma\left(g(\tilde{v}, x) + \phi(x)\zeta_j\right) \quad \text{with} \quad \zeta_1, \ldots, \zeta_k \sim \Pi^0_{w|Y}, \tag{6.15}$$

requiring only k Jacobian-vector products. In practice, for classification, we find k much smaller than the number of classes c suffices.

Algorithm 3: Sampling-based linearised hyperparameter learning and inference

Inputs: initial $a > 0$; $k, k' \in \mathcal{N}$, number of samples for stochastic EM and prediction, respectively.

Compute g-prior scaling vector s as in (6.12)

Sample random regularisers $w'_{0,1}, \ldots, w'_{0,k}$ per (6.14)

while a *has not converged* **do**

- Find posterior mode $\bar{w}$ by optimising linear model loss $\mathcal{L}(h(w, \cdot))$, given in (6.3)
- Draw posterior samples $\zeta_1 \ldots \zeta_k$ by optimising objective L' with $w'_{0,1}, \ldots, w'_{0,k}$
- Estimate effective dimension $\hat{\gamma}$, per (6.9), using samples $\zeta_1 \ldots \zeta_k$
- Update prior precision $a \leftarrow \hat{\gamma} / \|\bar{w}\|_2^2$

Sample k' random regularisers $w'_{0,1}, \ldots, w'_{0,k'}$ using optimised a

Draw corresponding posterior samples $\zeta'_1, \ldots, \zeta'_{k'}$ using (6.14)

Output: posterior samples $\zeta'_1, \ldots, \zeta'_{k'}$

6.2.5 Putting the pieces into a single algorithm for image classification

We now combine the methods described so far into a single algorithm that avoids storing Hessian H or covariance matrices H^{-1}, computing their log-determinants, or even instantiating Jacobian matrices $\phi(x)$, all of which have prevented the scalability of previous linearised Laplace implementations. We interact with NN Jacobians only through Jacobian-vector and vector-Jacobian products, which have the same asymptotic computational and memory costs as a NN forward-pass (Novak et al., 2022). Unless otherwise specified, we use the diagonal g-prior and a scalar regularisation parameter. Algorithm 3 summarises our method and Figure 6.5 shows an illustrative example.

An algorithm for image classification

Algorithm 4 provides a detailed procedure for applying our stochastic EM iteration to image classification while using g-prior feature scaling, described in (6.12). Therein, σ denotes the softmax function. The curvature of the softmax cross entropy loss at x_i, denoted B_i, is given by $B_i = \text{diag}[(p_i) - p_i p_i^T]$ for $p_i = \sigma(g(\tilde{v}, x_i))$ denoting our neural network's predictive probabilities. The notation $\odot$ refers to the elementwise product and to the elementwise power when used in an exponent.

The key hyperparameters of our algorithm are the number of samples to draw for the EM iteration, the number of EM steps to run, and SGD hyperparameters, namely learning rate, number of steps and batch-size. Empirically, we find that at most 5 EM steps are necessary for hyperparameter convergence and that as little as 1 sample can be used for

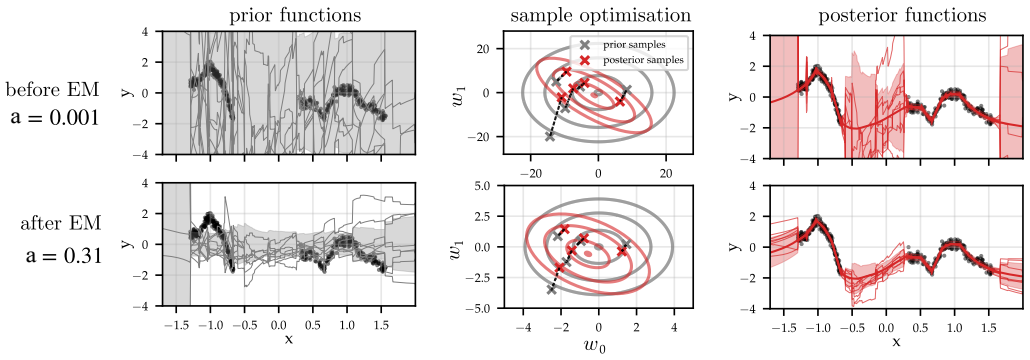


Fig. 6.5 Illustration of our procedure for a fully connected NN on the toy dataset of Antorán et al. (2020). Top: prior function samples present large std-dev. (left). When these samples are optimised (middle shows a 2D slice of weight space), the resulting predictive errorbars are larger than the marginal target variance (right). Bottom: after EM, the std-dev. of prior functions roughly matches that of the targets (left), the overlap between prior and posterior is maximised, leading to shorter sample trajectories (center), and the predictive errorbars are qualitatively more appealing (right).

the algorithm without degrading performance. Choosing SGD hyperparameters is more complicated. However, we are aided by the fact that lower loss values correspond to more precise posterior mean and sample estimates. As a result, we can tune these parameters on the train data, no validation set is required.

6.3 Demonstration: Image classification

We demonstrate our linear model inference and hyperparameter selection approach on the problem of estimating the uncertainty in NN predictions with the linearised Laplace method. First, in Section 6.3.1, we perform an ablation analysis on the different components of our algorithm using small LeNet-style CNNs trained on MNIST. In this setting, full-covariance Laplace inference (that is, exact linear model inference) is tractable, allowing us to evaluate the quality of our approximations. We then demonstrate our method at scale on CIFAR100 classification with a ResNet-18 (Section 6.3.2) and Imagenet with ResNet-50 (Section 6.3.3). We look at both marginal and joint uncertainty calibration and at computational cost.

6.3.1 Comparison with existing approximations on MNIST

We first evaluate our approach on MNIST $c{=}10$ class image classification, where exact linearised Laplace inference is tractable. The training set consists of $n{=}60k$ observations

Algorithm 4: Sampling-based linearised Laplace inference for image classification

Inputs: Linearised network h, unscaled feature expansion ϕ linearisation point $w_\star$, observations $x_1, \ldots, x_n$, negative log-likelihood function ℓ, initial precision $a > 0$, number of samples k

Define B_i**:**

$p_i \leftarrow \sigma(h(w_\star, x_i))$

return $\text{diag}(p_i) - p_i p_i^T$

for $j = 1, \ldots, k$ **do**

$w_j^0 \sim \mathcal{N}(0, a^{-1} I)$

$w'_j \leftarrow a^{-1} \sum_{i=1}^n \phi(x_i)^T \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0, B_i)$

$\zeta_j \leftarrow w_j^0$

$w_\star \leftarrow 0$

$s \leftarrow a^{-1} \left[\frac{1}{k} \sum_{j=1}^k {w'}_j^{\odot 2} \right]^{\odot -1/2}$

while a *has not converged* **do**

for $j = 1, \ldots, k$ **do**

$\zeta_j \leftarrow \text{SGD}_z \left(\|\Phi(s \odot z)\|_{\text{B}}^2 + a\|z - w_j^0 - (s \odot w'_j)\|_2^2, \text{ init}{=}\zeta_j \right)$

$w_\star \leftarrow \text{SGD}_w \left(\sum_{i=1}^n \ell(y_i, h((s \odot w), x_i)) + a\|w\|_2^2, \text{ init}{=}w_\star \right)$

$\hat{\gamma} \leftarrow \frac{1}{k} \sum_{j=1}^k \sum_{i=1}^n \|(\zeta_j \odot s)^T \phi(x_i)^T\|_{B_i}^2$

$a' \leftarrow \hat{\gamma} / \|w_\star\|_2^2$

for $j = 1, \ldots, k$ **do**

$w_j^0 \leftarrow \sqrt{\frac{a}{a'}} w_j^0$

$w'_j \leftarrow \frac{a}{a'} w'_j$

$a \leftarrow a'$

Output: Optimised precision a and weight samples $\zeta_1, \ldots, \zeta_k$

and we employ 3 LeNet-style CNNs of increasing size: "LeNetSmall" (d=14634), "LeNet" (d=29226) and "LeNetBig" (d=46024). The latter is the largest model for which we can store the covariance matrix on an A100 GPU. We draw samples and estimate posterior modes using SGD with Nesterov momentum. We use 5 seeds for each experiment, and report the mean and std. error.

Fidelity of sampling-based inference We compare our methods uncertainty using 64 SGD-based samples against approximate methods based on the NN weight point-estimate (MAP), a diagonal covariance, and against a KFAC estimate of the covariance (Martens and Grosse, 2015; Ritter et al., 2018) implemented with the `Laplace` library, in terms of similarity to the full-covariance lin. Laplace predictive posterior. As standard, we compute categorical predictive distributions with the probit approximation (Daxberger et al., 2021a). All methods use the same layerwise prior precision obtained with 5 steps of full-covariance

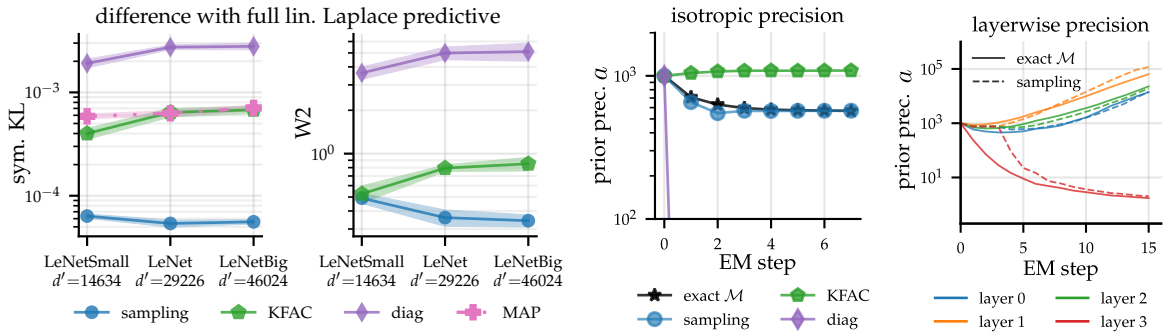


Fig. 6.6 Left: similarity to exact lin. Laplace predictions on the MNIST test-set, in terms of symmetric KL and Wasserstein-2 distance, for different approximate methods applied to NNs of increasing size. Centre right: comparison of EM convergence for a single hyperparameter across approximations. Right: layerwise convergence for exact and sampling methods.

EM iteration. The results are on the left hand side of Figure 6.6. For all three LeNet sizes, the sampled approximation presents the lowest categorical sym. KL and logit W2 distance to the exact lin. Laplace pred. posterior. The fidelity of competing approximations degrades with model size but that of sampling increases.

Accuracy of sampling hyperparameter selection We first compare our SGD sampling EM iteration with 16 samples to full-covariance EM on LeNet, both without the g-prior. Figure 6.6, middle-right, shows that for a single precision hyperparameter, both approaches converge in about 3 steps to the same value. In this setting, the diagonal covariance approximation diverges, and KFAC converges to a biased solution. We also consider learning layer-wise prior precisions by using the layerwise version of MacKay's M-step update. Here, neither the full covariance nor sampling methods converge within 15 EM steps. The precisions for all but the final layer grow in all steps. This reveals a pathology of this prior parametrisation: only the final layer's Jacobian, i.e. the final layer activations, are needed to accurately predict the targets; other features are pruned away.

We further compare the evidence approximations implied by a number of popular Laplace posterior approximations to the one from our SGD-based posterior samples. Figure 6.7 displays the evidence approximations obtained by plugging different posterior covariance approximations into the Laplace evidence given in (6.2). In particular, we consider the full-covariance Laplace evidence (denoted $\mathcal{M}$ in the plot), which we note does not match the exact model evidence due to the non-quadratic classification loss, the KFAC approximation to the covariance (labelled KFAC GGN), a single-sample KFAC Fisher estimate of the covariance (KFAC EF), the KFAC empirical Fisher matrix, and a diagonal Laplace covariance. We refer the reader to Daxberger et al. (2021a); Immer et al. (2021a) for a review of these

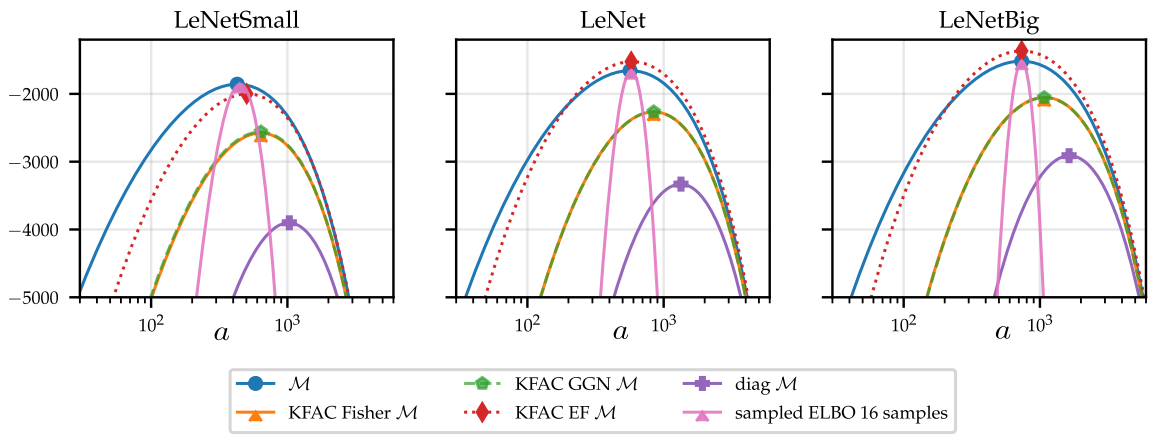


Fig. 6.7 Full covariance linearised Laplace evidence $\mathcal{M}$ together with approximations to this curve that rely on different covariance matrix approximations. A marker is placed at each curve's optima. We consider convolutional networks of increasing size (left to right) trained on the MNIST dataset.

approximations. We also include the linear-Gaussian ELBO given in (3.5) and discussed in Section 6.1.1, where the approximate posterior is given by 16 SGD-based samples. In all cases, we initialise the regulariser at an optima found by applying the EM algorithm while using the full covariance Laplace evidence $\mathcal{M}$ in the M-step. In this way, we may use the deviation of different objectives' optima from the optima of $\mathcal{M}$ as estimates of the bias in their corresponding approximations. The KFAC and KFAC-Fisher approximations result in a systematic overestimation of the evidence optima which grows with model size. This issue is even more pronounced for the diagonal covariance approximation. Surprisingly, we find the empirical Fisher to provide an accurate approximation. A similar finding is reported by (Immer et al., 2021a). This is surprising, given that the empirical Fisher is known to provide a heavily biased estimate of loss curvature and thus perform poorly for optimisation tasks (Kunstner et al., 2019). The sample-based ELBO shows close to no bias in its optima. This matches our experiments from Section 6.3.2, where the sample-based EM algorithm behaves well even when using very few samples.

6.3.2 Predictive performance and robustness on CIFAR-100

We showcase the stability and performance of our approach by applying it to CIFAR100 $c{=}100$-way image classification. The training set consists of $n{=}50k$ observations, and we employ a ResNet-18 model with $d \approx 11M$ parameters. To the best our knowledge, this is the first lin. Laplace approach that is capable of scaling to the CIFAR100 dataset, as the

high-parameter and high-output dimensions prove intractable even on modern hardware. Unless specified otherwise, we run 8 steps of EM with 6 samples to select a. We then optimise 64 samples to be used for prediction. We run each experiment with 5 different seeds reporting mean and std. error.

Stability and cost of sampling algorithm Figure 6.4 shows that our sample-based EM converges in 6 steps, even when using a single sample. At convergence, $a \approx 10^4$ and $\hat{\gamma} \approx 700$, so $2a\gamma = 2 \times 700 \times 10^4 = 1.4 \times 10^7 > 1.1 \times 10^7 = \operatorname{Tr} M$. Thus, (4.28) is satisfied and our low variance sample-then-optimise objective (4.17) presents better properties even at convergence. We use 50 epochs of optimisation for the posterior mode and 20 for sampling. When using 2 samples, the cost of one EM step with our method is 45 minutes on an A100 GPU; for the KFAC approximation, this takes 20 minutes.

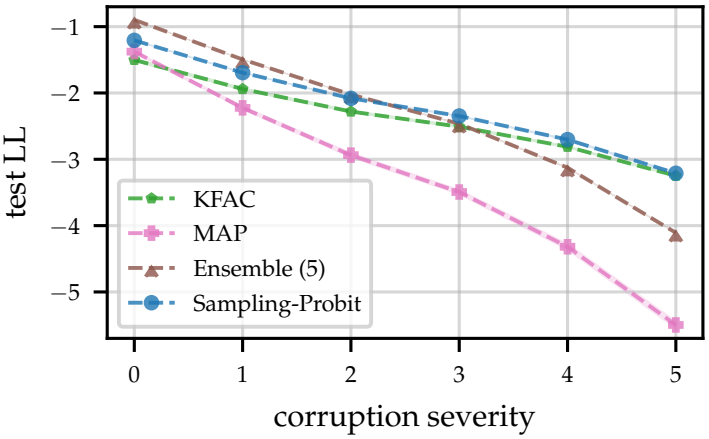


Fig. 6.8 Performance under distribution shift for ResNet-18 on CIFAR100.

Evaluating performance in the face of distribution shift We employ the standard benchmark for evaluating methods' test Log-Likelihood (LL) on the increasingly corrupted data sets of Hendrycks and Gimpel (2017); Snoek et al. (2019a). We compare the predictions made with our approach to those from deep ensembles, arguably the strongest baseline for uncertainty quantification in deep learning (Ashukha et al., 2020; Lakshminarayanan et al., 2017). We use a 5 element ensemble, as this is standard in the literature (Antorán et al., 2020; Daxberger et al., 2020) and the number of vector Jacobian products needed to train it roughly matches the amount used for 7 steps of sample-based EM optimisation with 6 posterior samples. We also consider a point-estimated predictions (MAP), and with a KFAC approximation of the lin. Laplace covariance (Ritter et al., 2018). For the latter,

constructing full Jacobian matrices for every test point is computationally intractable, so we use 64 samples for prediction, as we do for SGD sampling. The KFAC covariance structure leads to fast log-determinant computation, allowing us to learn layer-wise prior precisions (following Immer et al., 2021a) for this baseline using 10 steps of non-sampled EM. For both lin. Laplace methods, we use the standard probit approximation to the categorical predictive distribution (Daxberger et al., 2021b). Figure 6.8 shows that for in-distribution inputs, ensembles performs best and KFAC overestimates uncertainty, degrading LL even relative to point-estimated MAP predictions. Conversely, our method improves LL. For sufficiently corrupted data, our approach outperforms ensembles, also edging out KFAC, which fares well here due to its consistent overestimation of uncertainty.

	κ	MAP	Ensemble (5)	KFAC	Sampling
marginal LL	1	-1.40 $\pm$ 0.00	**-0.90 $\pm$ 0.00**	-1.12 $\pm$ 0.01	-1.07 $\pm$ 0.01
joint LL	2	-13.97 $\pm$ 0.01	-6.86 $\pm$ 0.01	**-4.92 $\pm$ 0.04**	-5.14 $\pm$ 0.04
	3	-27.89 $\pm$ 0.03	-14.17 $\pm$ 0.03	-10.83 $\pm$ 0.12	**-10.77 $\pm$ 0.09**
	4	-41.83 $\pm$ 0.03	-22.29 $\pm$ 0.04	-19.02 $\pm$ 0.22	**-18.04 $\pm$ 0.18**
	5	-55.89 $\pm$ 0.02	-31.07 $\pm$ 0.09	-29.40 $\pm$ 0.40	**-26.75 $\pm$ 0.26**

Table 6.1 Comparison of methods' marginal and joint prediction performance for ResNet-18 on CIFAR100.

Joint predictions Joint predictions are essential for sequential decision making, but are often ignored in the context of NN uncertainty quantification (Janz et al., 2019). To address this, we replicate the "*dyadic sampling*" experiment proposed by Osband et al. (2022). We group our test-set into sets of κ data points and then uniformly re-sample the points in each set until sets contain τ points. That is, multiple coppies of the κ original points. We then evaluate the LL of each set jointly. Since each set only contains κ distinct points, a predictor that models self-covariances perfectly should obtain an LL value at least as large as its marginal LL for all values of κ. We use τ=$10(\kappa - 1)$ and repeat the experiment for 10 test-set shuffles. Our setup remains the same as above but we use Monte Carlo marginalisation to push our Gaussian predictive distribution through the softmax instead of the probit approximation, since the latter discards covariance information. Table 6.1 shows that ensembles make calibrated predictions marginally but their joint predictions are poor, an observation also made by Osband et al. (2023). Our approach is competitive for all κ, performing best in the challenging large κ cases.

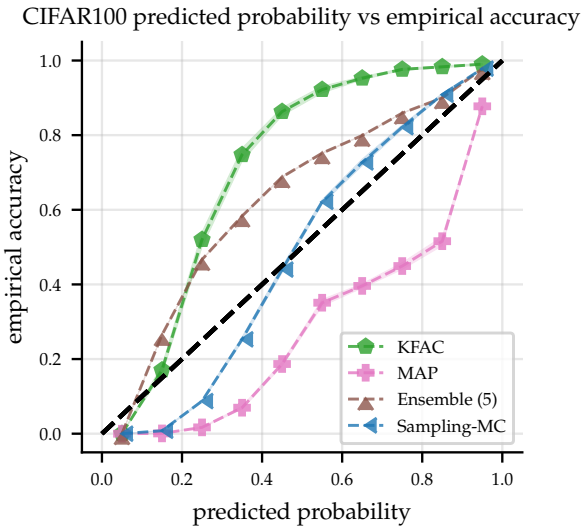


Fig. 6.9 Confidence vs accuracy plot (also known as a reliability diagram) for our CIFAR100 classification experiment.

Calibration of predictive uncertainty For the standard CIFAR100 test set, we separate our predicted probabilities into 10 equal width bins between 0 and 1. For each bin, we plot the proportion of targets that coincide with the class for which the predicted probability falls into the bin. This is shown in Figure 6.9. KFAC overestimates uncertainty at all confidence levels whereas MAP underestimates it. Both sample-based linearised Laplace and ensembling show significantly improved calibration. While ensembles show a small amount of uncertainty overestimation consistently, our method underestimates uncertainty for low predicted probabilities and overestimates it for large predicted probabilities.

6.3.3 Predictive performance on Imagenet

We demonstrate the scalability of our approach by applying it to Imagenet $c{=}1000$-way image classification (Russakovsky et al., 2015). The training set consists of $n{\approx}1.2M$ observations, and we employ a ResNet-50 model with $d \approx 25M$ parameters. To the best our knowledge, this is the first lin. Laplace approach that is capable of scaling to the Imagenet dataset, as the high-parameter and high-output dimensions prove intractable even on modern hardware. Our setup largely matches that described in Section 6.3.2 for CIFAR100 but we run 6 steps of EM with 6 samples to select a single regulariser parameter a. We then optimise 90 samples to be used for prediction. Our computational budget only allows for a single run. Thus, we do not provide errorbars.

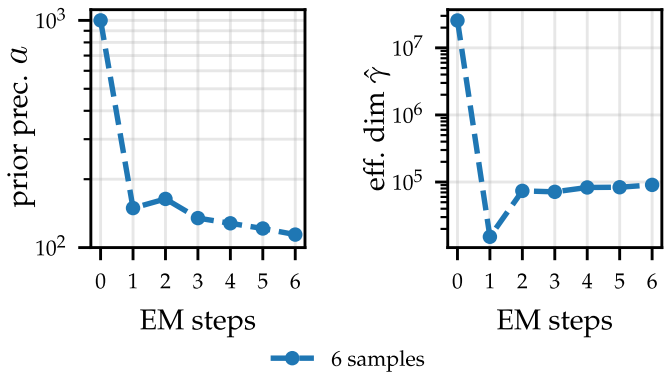


Fig. 6.10 Prior precision optimisation trajectories for ResNet-50 on Imagenet.

	κ	MAP	Ensemble 5 NNs	KFAC init	KFAC 5EM	Sampling 6EM	Sampling α=11.4
marginal LL	1	-0.936	**-0.815**	-1.449	-1.493	-0.924	-0.917
joint LL	2	-9.347	-6.700	-6.289	-6.286	-7.814	**-5.611**
	3	-18.733	-13.268	-12.112	-12.246	-15.065	**-10.675**
	4	-28.093	-20.029	-19.872	-20.493	-22.416	**-16.154**
	5	-37.416	-26.938	-29.839	-31.221	-29.787	**-21.981**

Table 6.2 Comparison of methods' marginal and joint predictive performance for ResNet-50 on Imagenet.

Stability and cost of sampling algorithm At each EM step, we run 10 epochs of optimisation to find the linear model's posterior mean and a single epoch of optimisation to draw samples. Each EM step takes roughly 26 hours on a TPU-v3 accelerator. Figure 6.10 shows the regularisation strength reaches values $\sim$150 in 1 step and then drifts slowly towards lower values without fully converging within 6 steps. This suggests the optimal a value may lie bellow 100. Unfortunately, we are unable to verify this, as the required computational cost exceeds our budget. Taking $a \approx 100$ and $\hat{\gamma} \approx 10^5$, we obtain $2a\gamma \approx \text{Tr}\ M = 2.5 \times 10^7$. According to (4.28) and thus our proposed sampling objective is expected to reduce variance throughout optimisation.

Marginal and joint predictions Using the same setup from the CIFAR100 joint prediction experiment above, but drawing 90 samples with our method and KFAC, we make marginal and joint predictions on the Imagenet test set. The results are shown in Table 6.2. Our methods regulariser optimisation trajectory in Figure 6.10 suggests a value lower than the one obtained after 6 EM steps ($a = 114$) may be preferred. Thus, we also report results with a value 10 times lower: $a = 11.4$. For KFAC, we optimise layerwise prior precisions using 5 EM steps. This leads to small precisions which produce underconfident predictions and poor results. We

attribute this to bias in the KFAC estimate of the covariance log-determinant. For comparison, we include KFAC results with a single regularisation parameter set to our initialisation value a=10000 (labelled "init"). This choice maintains or improves performance across κ values. Similarly to CIFAR100, ensembles obtains the strongest marginal test log-likelihood followed by our sampling approach for both regularisation strength values. KFAC overestimates uncertainty providing worse marginal performance than a single point-estimated network for both regularisation strength values. With a=11.4, our sampling approach performs best in terms of joint LL. Again, we find ensembles model joint-dependencies poorly. For κ values between 2 and 5, their performance is comparable to that of the KFAC approximation.

Concurrently with the present work, Deng et al. (2022) introduce "ELLA", a Nyström-based approximation to the Laplace covariance. With ResNet-50 on Imagenet, the authors report a marginal (κ=1) test LL of -0.948, which is worse than our MAP model. However, differences in the MAP solution upon which the Laplace approximation is built (theirs obtains -0.962 LL) make Deng et al. (2022)'s results not directly comparable with ours. ELLA does not provide a model evidence objective and thus Deng et al. (2022)'s result relies on validation-based tuning of the regularisation strength.

6.4 Discussion

This chapter introduced a sample-based approximation to inference and hyperparameter selection in Gaussian linear multi-output models. The approach is asymptotically unbiased, allowing us to scale the linearised Laplace method to ResNet models on CIFAR100 and Imagenet without discarding covariance information, which was computationally intractable with previous methods. The uncertainty estimates obtained through our method are well-calibrated not just marginally, but also jointly across predictions. Thus, our work may be of interest in the fields of active and reinforcement learning, where joint predictions are of importance, and computation of posterior samples is often needed.

With this, we have largely delivered on the goals of the thesis; scaling calibrated uncertainty estimation to real-world-sized models and datasets. Chapter 7, applies the methods developed so far to uncertainty estimation and experimental design for tomographic image reconstruction. Our scalable methods will allow us to perform uncertainty estimation for high-resolution volumetric reconstructions from neural networks, a problem not tackled before because of its large computational cost.

Since the publication of Antorán et al. (2023), which formed the basis for this chapter, there have been further efforts, using both Bayesian and non-Bayesian methods, to obtain calibrated uncertainty estimates from large NNs trained on large datasets. Most notable is the work of Osband et al. (2023), who use the sample-then-optimise objective (2.38) to draw samples of the weights of a small ad-hoc neural network placed on top of a pre-trained model's final layer activations. This procedure does not even approximately draw samples from the true posterior of the ad-hoc network's weights, but provides calibrated uncertainty estimates in practise, both in terms of marginal and joint predictions. Also worth mentioning is the work of Shen et al. (2024), which represents the latest effort to adapt a standard optimiser used in deep learning to learn the mean and variance vector of a mean field variational posterior.

Chapter 7

Uncertainty estimation and experimental design for computed tomography with the linearised deep image prior

Linear inverse problems in imaging aim to recover an unknown image $x \in \mathbb{R}^{d_x}$ from measurements $y \in \mathbb{R}^c$, which are often modelled by the application of a forward operator $\mathcal{T} \in \mathbb{R}^{c \times d_x}$ to the image, and the addition of Gaussian noise $\varepsilon \sim \mathcal{N}(0,\, b^{-1} I_c)$. That is

$$y = \mathcal{T}\, x + \varepsilon. \tag{7.1}$$

This acquisition model is ubiquitous in machine vision, computed tomography (CT), and magnetic resonance imaging, among other applications. Due to the inherent ill-posedness of the task (e.g. $c \ll d_x$), suitable regularisation, or prior assumptions, are crucial for the stable and accurate recovery of x (Ito and Jin, 2014; Tikhonov and Arsenin, 1977).

In this chapter, we focus on CT. Here, an emitter sends X-ray quanta through the object being scanned. The quanta are captured by d_p detector elements placed opposite the emitter. Each row of $\mathcal{T}$ tells us about which regions (pixels) the X-ray quanta will pass through before reaching a detector element. This is illustrated in Figure 7.1. The number of X-ray quanta measured by a detector pixel conveys information about the attenuation coefficient of the material present along the quanta's path. This procedure is repeated at $d_{\mathcal{B}}$ angles, yielding a measurement of dimension $c = d_p \cdot d_{\mathcal{B}}$, corresponding to the $c \times d_x$ sized linear operator $\mathcal{T}$, which is given by the discrete Radon transform.

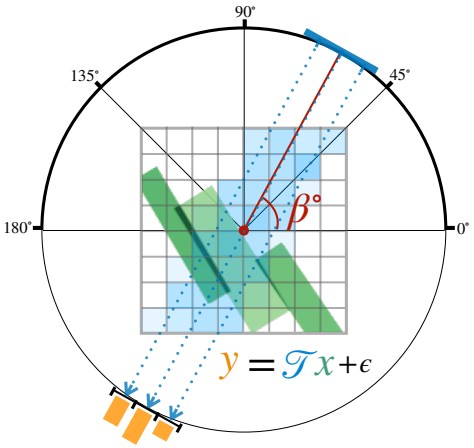


Fig. 7.1 A schematic diagram of 2D parallel beam CT geometry, used in our image reconstruction experiments. In the diagram, the detector is set to angle β. At this angle, a d_p dimensional observation is generated by the application of a $d_p \times d_x$ sized block of the $\mathcal{T}$ operator to the input x. In this plot, $d_p = 3$, $d_x = 64$ and the non-zero entries of the $d_p \times d_x$ sized block of $\mathcal{T}$ correspond to the pixels with blue colouring that the X-ray quanta pass through. We scan at $d_\mathcal{B}$ angles, generating a full $c = d_p d_\mathcal{B}$ dimensional observation.

In recent years, deep-learning based approaches have achieved outstanding performance on a wide variety of tomographic problems (Arridge et al., 2019; Ongie et al., 2020; Wang et al., 2020). Most deep learning methods are supervised; they rely on large volumes of paired training data. Alas, these often fail to generalise out-of-distribution (Antun et al., 2020); small deviations from the distribution of the training data can lead to severe reconstruction artefacts. Pathologies of this sort call for both unsupervised deep learning methods—free from training data and thus mitigating hallucinatory artefacts (Bora et al., 2017; Heckel and Hand, 2019; Tölle et al., 2021)—and uncertainty quantification (Kompa et al., 2021; Vasconcelos et al., 2022)—informing the user about (un)reliability in reconstructions.

We focus on the deep image prior (DIP), perhaps the most widely adopted unsupervised deep learning approach (Ulyanov et al., 2018a). DIP regularises the reconstructed image $\tilde{x}$ by reparametrising it as the output of a deep convolutional neural network (CNN). It does not require paired training data, relying solely on the structural biases induced by the CNN architecture. The DIP has proven effective on tasks ranging from denoising and deblurring to challenging tomographic reconstructions (Baguer et al., 2020; Barutcu et al., 2022; Cui et al., 2021; Darestani and Heckel, 2021; Gong et al., 2019; Knopp and Grosser, 2021; Liu et al., 2019). Nonetheless, the DIP only provides point reconstructions without uncertainty estimates.

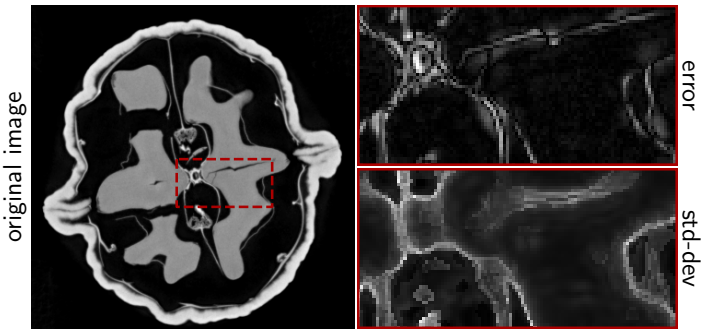


Fig. 7.2 X-ray reconstruction (501×501 px^2) of a walnut (left), the absolute error of its CT reconstruction (top) and pixel-wise uncertainty from the linearised DIP (bottom).

In this chapter, we apply the methods developed through this thesis to equip DIP reconstructions with reliable uncertainty estimates. In literature, there are two notable probabilistic reformulations of the DIP (Cheng et al., 2019; Tölle et al., 2021), but their focus is on preventing overfitting rather than accurately estimating uncertainty. Distinctly from these, we only estimate the uncertainty associated with a specific reconstruction. We do this by computing Gaussian-linear model type error-bars for a local linearisation of the DIP around its mode (Immer et al., 2021a; Khan et al., 2019a; Mackay, 1992a), and refer to the method as *linearised DIP*. Linearised approaches have recently provided state-of-the-art uncertainty estimates for supervised deep learning models (Daxberger et al., 2021b). We also explore the incorporation of the total variation (TV) regulariser, ubiquitous in CT reconstruction, as a Bayesian prior for the weights of the linearised model. This regulariser is unnormalised and does not lend itself to standard Laplace (i.e. local Gaussian) approximations (Helin et al., 2022a). We tackle this issue using predictive complexity prior (PredCP) framework of Nalisnick et al. (2021).

We demonstrate our approach on high-resolution CT reconstructions of real-measured 2D and 3d Micro CT (μCT) projection data. An example of the former is in Figure 7.2. Empirically, the method's pixel-wise uncertainty estimates predict reconstruction errors more accurately than existing approaches to uncertainty estimation with the DIP. This is not at the expense of accuracy in reconstruction: the reconstruction obtained using the standard regularised DIP method (Baguer et al., 2020) is preserved as the predictive mean, ensuring compatibility with advancements in DIP research.

We then go on to leverage the aforementioned uncertainty estimates to perform adaptive experimental design for CT scan angle selection. We consider a setting where the CT scan is performed in two phases. First, a sparse pilot scan is performed to provide data with which to fit adaptive methods. These are then used to adaptively select angles for a full scan using the

linearised deep image prior as a data-dependent prior. We demonstrate this procedure with a synthetic dataset where a different "preferential" angle is most informative for each image. Unlike simple linear models, the linearised DIP's designs depend on previously observed targets. This adaptivity allows linearised DIP designs to outperform the equidistant angle baseline, which is almost always used in deployment.

The contributions of this chapter can be summarised as follows.

- We propose a novel approach to bestow reconstructions from the TV-regularised DIP with uncertainty estimates, by linearising the DIP around its optimised reconstruction and providing the linear model's error-bars as a surrogate for those of the DIP. We perform sample-based EM inference in this model, scaling to high resolution real-measured 2d reconstructions and 3d volumetric reconstructions. To be best of our knowledge, this is the first instance of uncertainty estimation for NN-based 3d volumetric CT reconstruction. Our approach yields far more accurate uncertainty estimation than existing probabilistic formulations of the DIP.
- We leverage the linearised DIP as a data-dependent prior for CT experimental design. This allows us to perform adaptive design, where successive acquisition locations dependent on previous observed targets (as opposed to only the input locations), while preserving the tractability of a linear model. This method outperforms equidistant angle selection on a synthetic task.

The rest of this chapter is organised as follows. Section 7.1 covers the DIP, the TV regulariser and other preliminaries not introduced earlier in the thesis. Section 7.2 presents the linearised DIP and a novel TV-based prior for the linear model's parameters. Section 7.3 discusses efficient approaches to inference. Section 7.4 presents a demonstration of the linearised DIP on real-measured high-resolution μCT data. Section 7.5 introduces experimental design with linear models and discusses how the linearised DIP can be used within this framework. Finally, Section 7.6, demonstrates our approach to experimental design on synthetic data, and Section 7.7 concludes the chapter.

7.1 Preliminaries

This section reviews some CT-specific concepts that were not covered in earlier chapters of the thesis.

7.1.1 Total variation regularisation

The imaging problem, given in (7.1), admits a linear subspace of solutions consistent with the observation y[1]. Thus, regularisation is needed for stable reconstruction. Total variation (TV) is perhaps the most well established regulariser (Chambolle et al., 2010; Rudin et al., 1992). The anisotropic TV semi-norm of an image vector $x \in \mathbb{R}^{d_x}$ imposes an L_1 constraint on image gradients:

$$\mathrm{TV}(x) = \sum_{i,j} |\mathfrak{x}_{i,j} - \mathfrak{x}_{i+1,j}| + \sum_{i,j} |\mathfrak{x}_{i,j} - \mathfrak{x}_{i,j+1}|, \tag{7.2}$$

where $\mathfrak{x} \in \mathbb{R}^{\mathfrak{h} \times \mathfrak{w}}$ denotes the vector x reshaped into an image of height $\mathfrak{h}$ by width $\mathfrak{w}$, and $d_x = \mathfrak{h} \cdot \mathfrak{w}$. This leads to the regularised reconstruction formulation

$$\tilde{x} \in \operatorname*{arg\,min}_{x \in \mathbb{R}^{d_x}} \mathcal{L}(x) \quad \text{with} \quad \mathcal{L}(x) \coloneqq b \| \mathcal{T}\, x - y \|_2^2 + \mathrm{TV}(x), \tag{7.3}$$

where the hyperparameter $b > 0$ determines the strength of the regularisation relative to the fit term.

7.1.2 Bayesian inference for inverse problems

The Bayesian framework provides a consistent approach to uncertainty estimation in imaging problems (Kaipio and Somersalo, 2005; Seeger and Nickisch, 2011; Stuart, 2010). The image to be recovered is treated as a random variable. Instead of finding a single best reconstruction $\tilde{x}$, we aim to find a posterior $\mathcal{P}_{x|y}$, with density $\rho(x|y)$ that scores every candidate $x \in \mathbb{R}^{d_x}$ according to its agreement with the observation y and prior density $\rho(x)$. The loss in (7.3) can be viewed as the negative log of an unnormalised posterior density, i.e. $\rho(x|y) \propto \exp(-\mathcal{L}(x))$, and $\tilde{x}$ as its mode, i.e. the *maximum a posteriori* (MAP) estimate. The least squares loss corresponds to a Gaussian likelihood $p(y|x) = \mathcal{N}(y;\, \mathcal{T}\, x, I)$ and the TV regulariser to a prior over images $\mathcal{P}$ with density $\rho(x) \propto \exp(-\lambda\, \mathrm{TV}(x))$. With this, we are ready to crank the lever of Bayesian reasoning, as introduced in Section 2.1.2.

Our work partially departs from this framework in that it *solely concerns itself with characterising plausible reconstructions around the mode* $\tilde{x}$ (Mackay, 1992a). This has two key advantages, 1) *tractability*: the likelihood induced by NN reconstructions is strongly multi-modal, and both analytically and computationally intractable. In contrast, the posterior

[1]This statement disregards the effects of the observation noise, which introduces a strictly convex constraint, but in practise it is very weak.

for the local model is Gaussian; 2) *interpretablity*: even if we could obtain the full posterior, downstream stakeholders not versed in probability are likely to have little use for it. A single reconstruction and its pixel-wise uncertainty may be more interpretable to end-users (Antorán et al., 2021; Bhatt et al., 2021).

7.1.3 The Deep Image Prior (DIP)

The DIP (Ulyanov et al., 2018a, 2020) reparametrises the reconstructed image as the output of a CNN $g : \mathbb{R}^d \to \mathbb{R}^{d_x}$ with learnable parameters $v \in \mathbb{R}^d$ and a fixed input, which we have omitted from our notation for clarity. The DIP can be seen as a reparametrisation of the reconstructed image that provides a favourable structural bias. We introduce the optimisation problem

$$\tilde{v} \in \arg\min_{v \in \mathbb{R}^d} b\| \mathcal{T}\, g(v) - y\|_2^2 + \mathrm{TV}(g(v)), \tag{7.4}$$

and the recovered image is given by $\tilde{x} = g(\tilde{v})$. Penalising the TV of the DIP's output avoids the need for early stopping and improves reconstruction fidelity (Baguer et al., 2020; Liu et al., 2019). The standard choice of CNN architecture is the fully convolutional U-net (Ronneberger et al., 2015). We also adopt this architecture in this chapter. Although the parameters v must be optimised separately for each new measurement y, we follow (Barbano et al., 2022c; Knopp and Grosser, 2021) to reduce the cost with task-agnostic pretraining.

Since its introduction by Ulyanov et al. (2018a, 2020), the DIP has been improved with early stopping (Wang et al., 2021), TV regularisation (Baguer et al., 2020; Liu et al., 2019), and pretraining (Barbano et al., 2023, 2022c; Knopp and Grosser, 2021). We build upon these recent advancements by providing a scalable method to estimate the error-bars of DIP's reconstructions. This is a relatively unexplored topic. Building upon Garriga-Alonso et al. (2019) and Novak et al. (2019), Cheng et al. (2019) show that in the infinite-channel limit, the DIP converges to a Gaussian process (GP). In the finite-channel regime, the authors approximate the posterior distribution over the DIP's parameters with stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011). Laves et al. (2020) and Tölle et al. (2021) use factorised Gaussian variational inference (Blundell et al., 2015) and MC dropout (Hron et al., 2018; Vasconcelos et al., 2022), respectively. These probabilistic treatments of DIP primarily aim to prevent overfitting, as opposed to accurately estimating uncertainty. While they can deliver uncertainty estimates, their quality tends to be poor. In fact, obtaining reliable uncertainty estimates from deep-learning based approaches, like the DIP, largely remains a challenging open problem (Antorán, 2019; Antorán et al., 2020; Ashukha et al., 2020; Barbano et al., 2022a; Foong et al., 2020; Snoek et al., 2019b).

7.2 Linearised DIP uncertainty estimation for CT

In this section, we build a probabilistic model to characterise the uncertainty associated with reconstructions around $\tilde{v}$, a mode of the regularised DIP objective obtained using (7.4). Section 7.2.1 describes the construction of a linearised surrogate for the DIP reconstruction. Section 7.2.2 describes how to compute the surrogate model's error-bars and use them to augment the DIP reconstruction. Section 7.2.3 discusses how we include the effects of TV regularisation into the surrogate model.

7.2.1 From a prior over parameters to a prior over images

After training the DIP to an optimal TV-regularised setting $\tilde{x} = g(\tilde{v})$ using (7.4), we linearise the network around $\tilde{v}$ by applying (3.33), and obtain the affine in $w \in \mathbb{R}^d$ function $h(w)$. The error-bars obtained from Bayesian inference with $h(w)$ will tell us about the uncertainty in $\tilde{x}$. To this end, consider the Bayesian model,

$$\begin{aligned} y|w \sim \mathcal{N}(\mathcal{T}\, h(w), b^{-1}I), \quad w \sim \mathcal{N}(0, A^{-1}) \\ \text{with} \quad h(w) \coloneqq g(\tilde{v}) + \phi(w - \tilde{v}), \end{aligned} \tag{7.5}$$

where $\phi = \partial_v g(\tilde{v}) \in \mathbb{R}^{c \times d}$ is the Jacobian of our NN[2]. We will select the precision A to incorporate TV constraints into the computed error-bars in Section 7.2.3. We have introduced the noise variance b^{-1} as an additional hyperparameter which we will learn using the marginal likelihood.

Demonstration: sampling from the linearised DIP prior

To provide intuition about the linearised model, we push samples from $w \sim \mathcal{N}(0, A^{-1})$, through h. The resulting reconstruction samples are drawn from a Gaussian distribution with covariance $K \in \mathbb{R}^{d_x \times d_x}$ given by $\phi A^{-1} \phi^\top$. We show an example in Figure 7.3. Here, the Jacobian ϕ introduces structure from the NN function around the linearisation point $\tilde{v}$. For this example, we train our NN on CT data simulated by using the KMNIST dataset as the original images. Thus, out prior samples contain features from the KMNIST character that the DIP was trained on.

[2]In this chapter, ϕ is a matrix, as opposed to a function that returns a matrix, because our NN's input is clamped to a constant.

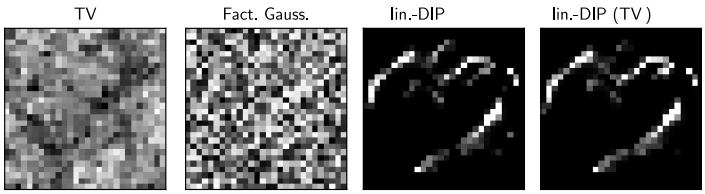


Fig. 7.3 Samples from different priors over the reconstructed image x. From left to right, the plots show samples from the TV prior with density $\propto \exp(-\mathrm{TV}(x))$, drawn with HMC, from an isotropic Gaussian prior, from a linearised DIP trained on a MNIST character, and from the same model but paired with the TV-PredCP prior over the weights introduced in Section 7.2.3. The latter leads to smoother samples with less artefacts than the standalone linearised DIP prior.

7.2.2 Computing the predictive uncertainty

We augment the DIP reconstruction $\tilde{x}$ with Gaussian predictive error-bars computed with the linearised model h described in (7.5). This yields the predictive distribution $\mathcal{N}(\tilde{x}, K_{x|y})$. Denoting the reconstruction space kernel matrix $K = \phi A^{-1}\phi^\top \in \mathbb{R}^{d_x \times d_x}$, the observation space prior covariance $K_{yy} = TKT^\top \in \mathbb{R}^{c\times c}$, and the cross terms $K_{xy} = KT^\top \in \mathbb{R}^{d_x \times c}$, the posterior covariance $K_{x|y} \in \mathbb{R}^{d_x \times d_x}$ is given by

$$K_{x|y} = \phi(b^2\phi^\top \mathcal{T}^\top \mathcal{T}\, \phi + A)^{-1}\phi^\top = K - K_{xy}(K_{yy} + b^{-1}I)^{-1}K_{xy}^\top. \tag{7.6}$$

Importantly, (7.6) depends on the inverse of the observation space covariance $K_{yy} + b^{-1}I$, which we expect to be much lower dimensional than the covariance over reconstructions, or parameters. Thus, the cost of computing (7.6) scales as $\mathcal{O}(d_x c^2)$.

7.2.3 Incorporating TV-smoothness into the prior over the weights

This section aims to design a prior that constraints h's error-bars, such that the model only considers low TV reconstructions as plausible. Our architecture if fully convolutional. We follow the guiding intuition that if the CNN's filters are smooth, its output will be so as well. With this, we place a block-diagonal Matérn-$1/2$ covariance Gaussian prior on the linearised model's weights, similarly to Fortuin et al. (2021). In particular, we introduce dependencies between parameters in the same CNN filter by constructing A as a block diagonal matrix. We denote the block corresponding to each filter as A_k, where k indexes the filter. These

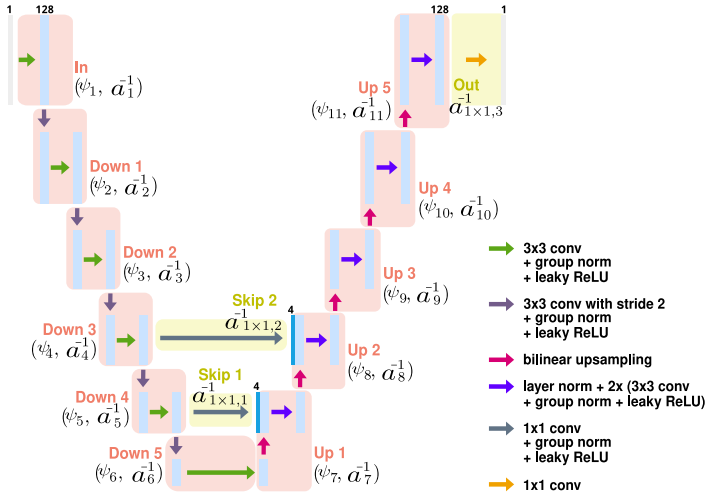


Fig. 7.4 A schematic of the U-net architecture used in our 2d μCT experiments experiments. For KMNIST, we use a reduced, 3-scale U-net without group norm layers. Each light-blue rectangle corresponds to a multi-channel feature map. We highlight the architectural components corresponding to each block $1, \ldots, D$ for which a separate prior is defined with red and yellow boxes.

matrix blocks are given by

$$[A_k^{-1}]_{ij,i'j'} = a_k^{-1} \exp\left(\frac{-\sqrt{(i-i')^2 + (j-j')^2}}{\psi_k}\right), \tag{7.7}$$

where the tuple (i, j) indexes the spatial location of a specific filter pixel in terms of height and width. The per-filter lengthscale ψ_k regulates the filter smoothness. The hyperparameter a_k^{-1} determines the marginal prior variance for each filter. Both parameters are shared among all filters in an architectural block in the U-net, indexed by $k \in \{1, 2, \ldots, r\}$. We write $\psi = [\psi_1, \psi_2, \ldots, \psi_r]$ and $a^{-1} = [a_1^{-1}, a_2^{-1}, \ldots, a_r^{-1}]$. A diagram of our U-net architecture that highlights all architectural blocks is provided in Figure 7.4. The chosen U-net architecture is fully convolutional and thus (7.7) applies to all parameters, reducing to a diagonal covariance for 1×1 convolutions.

In Figure 7.5, we experimentally verify that an image generated from a linearised NN prior with smoother filters will present lower TV. In particular, we find a bijective relationship between the each filter's lengthscale ψ_k and the expected TV $\mathbb{E}_{w_k \sim \mathcal{N}(0, A_k^{-1})}[\mathrm{TV}(\phi w)]$ where w_k is a sub-vector of w and the other filter's parameters (non w_k) are held fixed. This suggests we may use the predictive complexity prior (PredCP) framework of Nalisnick et al. (2021) to construct a prior over the parameters which acts as a surrogate for the TV prior. In particular,

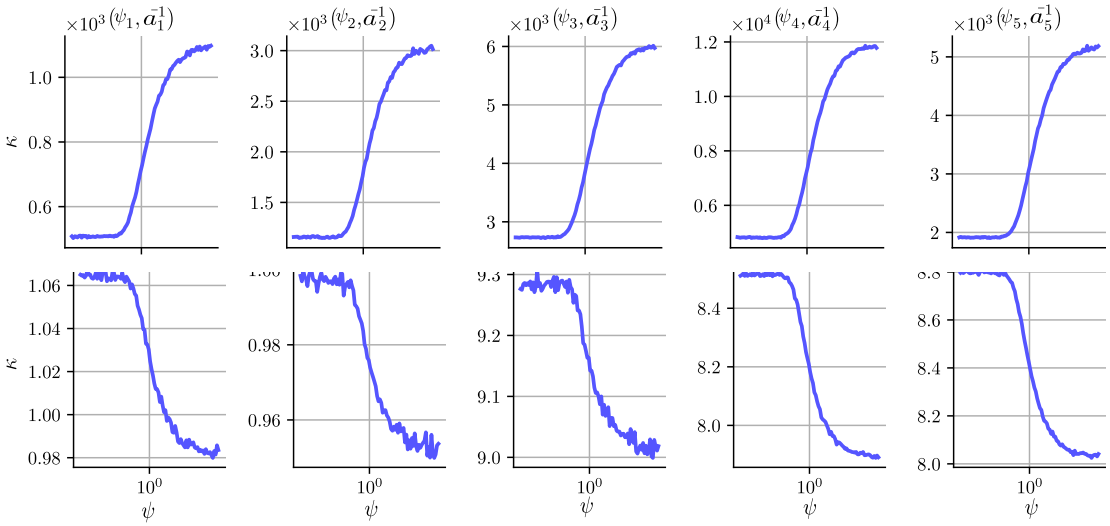


Fig. 7.5 Experimental evidence of the monotonicity (and thus invertibility) of the relationship between a CNN block's lengthscale ψ_k and the expected TV $\kappa = \mathbb{E}_{w_k \sim \mathcal{N}(0, A_k^{-1})}[\mathrm{TV}(\phi w)]$, computed across 50 linearised U-nets trained on different the KMNIST images. The horizontal axis represents lengthscale $\psi \in [0.01, 100]$. κ is estimated with 10k Monte Carlo samples. In the bottom row we scale the marginal variances of $JA^{-1}J^\top$ to be 1 for every value of ψ. This decouples ψ from a^{-1}, allowing us to observe the smoothing effect from larger lengthscales.

we construct a prior over the lengthscale parameters as:

$$\prod_{k=1}^{r} \mathrm{Exp}(\kappa_k) \left| \frac{\partial \kappa_k}{\partial \psi_k} \right|, \tag{7.8}$$

$$\text{with } \kappa_k := \mathbb{E}_{w_k \sim \mathcal{N}(0, A_k^{-1}) \prod_{i=1, i \neq k}^{r} \delta(w_i)} \left[\mathrm{TV}(\phi w) \right] \tag{7.9}$$

where the subscript $_k$ indicates we select the subvector of weights corresponding to CNN block k. We have related a block's contribution to the expected TV, κ_k, to the block's filter lengthscale ψ_k via the change of variables formula. The independence across blocks assumed in (7.8) ensures dimensionality preservation, formally needed in the change of variables. It follows from the triangle inequality that $\sum \kappa_k$ is an upper bound on the expectation under the distribution $\mathbb{E}_{w \sim \mathcal{N}(0, A^{-1})}[\mathrm{TV}(\phi w)]$, further motivating the factorisation.

Derivation Factorising yields an upper bound on the expected TV

Let $\mathcal{S}$ be the set of indices for all adjacent pixel pairs in an image. These images are flattened into d_x length vectors and thus can be indexed by a single number. We denote

by $\phi_j \in \mathbb{R}^d$ the row of the Jacobian ϕ corresponding to pixel i. We denote by ϕ_{jk} the Jacobian row subvector corresponding to pixel i and weights in NN block k. With this

$$\begin{aligned}
\mathbb{E}_{w\sim\mathcal{N}(0,A^{-1})}\left[\mathrm{TV}(\phi w)\right] &= \sum_{(i,j)\in\mathcal{S}} \mathbb{E}_{w\sim\mathcal{N}(0,A^{-1})}|(\phi_i w - \phi_j w)| \\
&= \sum_{(i,j)\in\mathcal{S}} \mathbb{E}_{w_k\sim\mathcal{N}(0,A^{-1})}|\sum_{k}^{r}(\phi_{ik} - \phi_{jk})w_k)|] \\
&\leq \sum_{(i,j)\in\mathcal{S}} \sum_{k}^{r} \mathbb{E}_{w_k\sim\mathcal{N}(0,A_k^{-1})}\left[|(\phi_{ik} - \phi_{jk})w_k|\right] \\
&= \sum_{k}^{r} \mathbb{E}_{\mathcal{N}(0,A_k^{-1})\prod_{i=1,i\neq k}^{r}\delta(w_i)}\left[\sum_{(i,j)\in\mathcal{S}}|(\phi_i - \phi_j)w|\right] \\
&= \sum_{k}^{r}\kappa_k,
\end{aligned}$$

Thus, the separable form of the TV prior as a regulariser ensures that the expected TV under the joint distribution of parameters is also regularised.

Note that (7.8) can be computed analytically. However, its direct computation is costly and we instead rely on numerical methods, described in Section 7.3.1. In Figure 7.3 we show samples from the linearised NN model where ψ is chosen using the marginal likelihood with TV-PredCP constraints. Incorporating the TV-PredCP leads to smoother samples with less discontinuities.

7.3 Approaches to scalable inference and hyperparameter learning

In a typical tomography setting, the dimensionality d_x of the image x and c of the observation y can be large, e.g. $d_x > 1\mathrm{e}5$ and $c > 5\mathrm{e}3$. Thus holding the input space covariance matrices (e.g. K and $K_{x|y}$) in memory is infeasible. This also complicates computing determinants, needed to evaluate Gaussian densities, and to learn hyperparameters. Following Chapter 6, we develop a series of approaches that avoid instantiating these matrices explicitly. We only access Jacobian and covariance matrices through matrix–vector products.

Section 7.3.1 introduces a hyperparameter learning objective that combines the linearised model's evidence with the TV-PredCP prior over filter lengthscales. We approximate the objective's gradients with CG. Section 7.3.2 discusses the computation of a randomised preconditioner for CG. Section 7.3.3 discards the TV-PredCP prior in favour of the g-prior. This allows us to employ the sample-based EM iteration from Chapter 6, in combination with CG, to accelerate inference. Section 7.3.4 discusses the extension of the latter algorithm to very large 3d volumetric reconstructions by substituting CG solves with SGD (as suggested in Chapter 4). Finally, we discuss making sample-based predictions that model covariances between pixels in Section 7.3.5

7.3.1 Conjugate-gradient hyperparameter learning for the PredCP TV prior

In this subsection we consider hyperparameter learning with the TV-PredCP prior introduced in Section 7.2.3. Here, the prior precision A is parametrised in terms of the vectors of block-wise marginal variances $a^{-1} \in \mathbb{R}^d$ and block-wise lengthscales $\psi \in \mathbb{R}^d$. To learn ψ, we combine the above objective with the TV-PredCP's log-density, which acts as a regulariser. The resulting expression used to learn the full set of hyperparameters (b^{-1}, a^{-1}, ψ) resembles a Type-II MAP (Williams and Rasmussen, 2006) objective

$$\begin{aligned}\log\, p(y|\psi;\, b^{-1}, a^{-1}) + \log p(\psi; a^{-1}) \\ \approx -\frac{1}{2} b||y - \mathcal{T}\, g(\tilde{v})||_2^2 - \frac{1}{2} w_\star^\top A w_\star - \frac{1}{2}\log|K_{yy} + b^{-1} I| \\ - \sum_{k=1}^{r} \kappa_k + \log\left|\frac{\partial \kappa_k}{\partial \psi_k}\right| + C,\end{aligned} \tag{7.10}$$

where C is independent of the hyperparameters and the vector $w_\star \in \mathbb{R}^{d_w}$ is the posterior mean of the linear model's parameters (see Section 5.2). We compute it as

$$w_\star = A^{-1}\phi^\top \mathcal{T}^\top (K_{yy} + b^{-1} I)^{-1} \left(y + \mathcal{T}\left(\phi\tilde{v} - g(\tilde{v})\right)\right) \tag{7.11}$$

and solve the therein contained linear system with CG. The vector we solve against consists of the observations offset by the constant in w terms in the tangent linear model (7.5).

The remaining bottleneck in evaluating (7.10) is the log-determinant $\log|K_{yy} + b^{-1} I|$, which has a cost $\mathcal{O}(c^3)$. Alas, we cannot apply the sample-based MacKay update from Section 6.2.1 to learn hyperparameters other than the entries of a diagonal prior precision

matrix. Thus, we resort to gradient descent with CG-based log-determinant gradient trace estimation, as described in Section 3.2.1. We use a preconditioner, which we describe in Section 7.3.2. Despite this, the large computational cost associated with this method only allows us to perform a single EM step. We summarise the procedure in algorithm 5. We go on to describe efficient estimation of the TV-PreCP term gradients.

Algorithm 5: Linearised deep image prior PredCP-TV inference

Inputs: noisy measurements y, a CNN $g(\cdot)$, operator $\mathcal{T}$, initial prior precision A.

1 $\tilde{v} \leftarrow$ `fit_DIP(`$\mathcal{T}, y, g(v)$`) // by minimising (7.4)`

2 $w_\star \leftarrow$ `fit_linearised_model(`$\mathcal{T}, y, g(\tilde{v})$`) // using (7.11)`

3 $\mathcal{P} \leftarrow$ `compute_preconditioner(` $\mathcal{T}, g(\tilde{v}), A$ `) // following Section 7.3.2.`

4 $\sigma_y^2, \{a_k^{-1}, \psi_k\}_{k=1}^r \leftarrow$ `optimise_hyperparams(`$\mathcal{T}, y, g(\tilde{v}), w_\star, \mathcal{P}$`) // with (7.10), and eqs. (7.12) and (7.13). Use preconditioned CG.`

5 $\hat{K}_{x|y} \leftarrow$ `fast_sampling(`$\mathcal{T}, g(\tilde{v}), \sigma_y^2, \{a_k^{-1}, \psi_k\}_{k=1}^D, \mathcal{P}$`) // following Section 7.3.5 and with preconditioned CG.`

Output: mean reconstruction $g(\tilde{v})$, posterior covariance estimate $\hat{K}_{x|y}$

MC sampling for TV-PredCP optimisation

For large images, exact evaluation of the expected TV with (7.9) is computationally intractable. Instead, we estimate the gradient of κ_k with respect to $\theta = (\sigma^2, \psi)$ using a Monte-Carlo approximation of the expectation

$$\frac{\partial \kappa_k}{\partial \theta} = \mathbb{E}_{w_k \sim \mathcal{N}(0, A_k^{-1})} \left[\frac{\partial \, \mathrm{TV}(x)}{\partial x} \phi_k \frac{\partial w_k}{\partial \theta} \right], \tag{7.12}$$

where $\phi_k = \frac{\partial g(v)}{\partial v_k}|_{v=\tilde{v}}$. $\frac{\partial \, \mathrm{TV}(x)}{\partial x}$ is evaluated at the sample $x{=}\phi_k w_k$ and $\frac{\partial w_k}{\partial \theta}$ is the reparametrisation gradient for w_k, a prior sample of the weights of CNN block k. The gradient for the Jacobian log-determinant term is $\frac{\partial}{\partial_\theta} \log \left| \frac{\partial \kappa_k}{\partial \theta} \right| = \frac{\partial^2 \kappa_k}{\partial \theta^2} / \frac{\partial \kappa_k}{\partial \theta}$, and since the second derivative of the TV semi-norm is almost everywhere zero, we have

$$\frac{\partial^2 \kappa_k}{\partial \theta^2} = \mathbb{E}_{w_k \sim \mathcal{N}(0, A_k^{-1})} \left[\frac{\partial \, \mathrm{TV}(x)}{\partial \theta} \phi_k \frac{\partial^2 w_k}{\partial \theta^2} \right]. \tag{7.13}$$

Demonstration: blockwise lengthscale optimisation

In Figure 7.6, demonstrate the use of (7.10) to learn the full set of prior hyperparameters depicted in Figure 7.4. We also ablate the TV-PredCP regulariser to better understand its

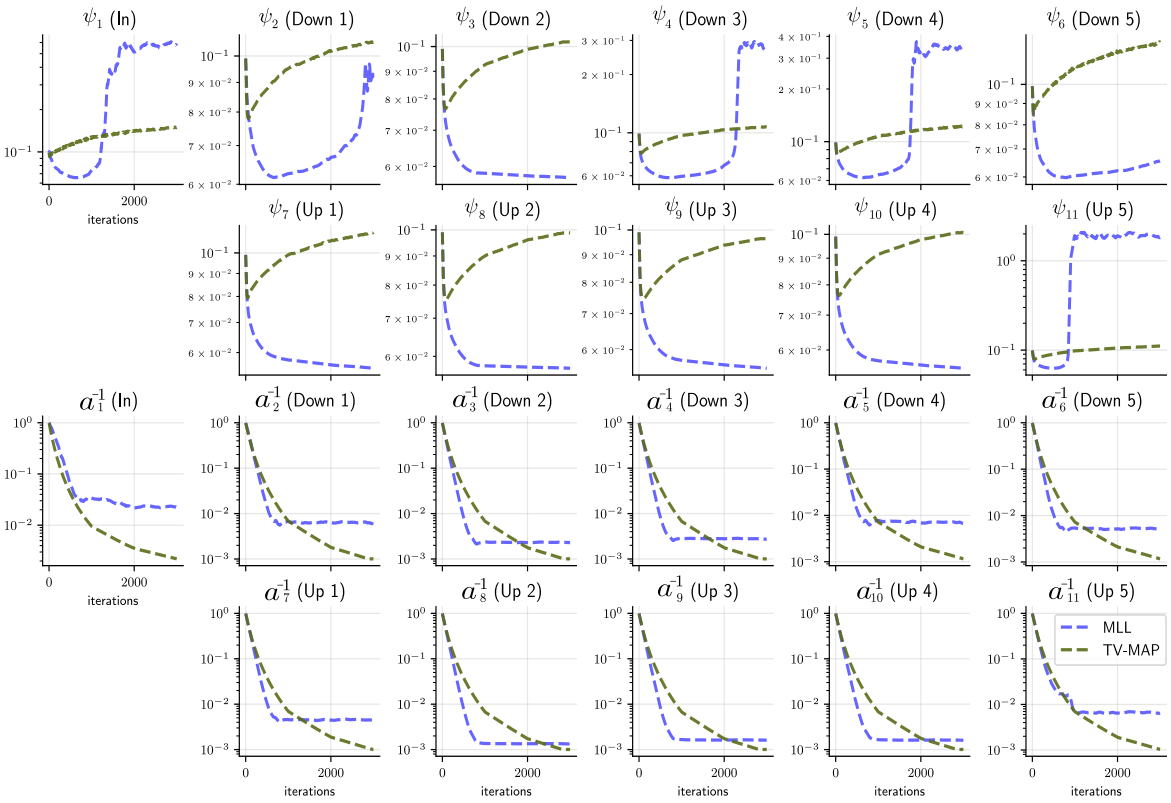


Fig. 7.6 Optimisation traces for the lengthscales and marginal variances corresponding to our U-net's 3×3 convolution layers. We consider both MLL and Type-II MAP and we use the Walnut data described in Section 7.4. The TV-PredCP leads to larger prior lengthscales ψ and lower variances a^{-1}.

effects. We refer to the ablated setting as "MLL" and the setting where the TV-PredCP is kept as "Type-II MAP". We use the high-resolution real-measured dataset of Der Sarkissian et al. (2019) and provide full details on the experimental setup in Section 7.4.1.

During MLL and Type-II MAP optimisation, many layers' prior variance goes to $a^{-1} \approx 0$. This phenomenon is known as "automatic relevance determination" (Mackay, 1996; Tipping, 2001), and simplifies our linearised network, preventing uncertainty overestimation. Type-II MAP hyperparameters optimisation drives ψ to larger values, compared to MLL. This restricts the linearised DIP prior, and thus the induced posterior, to functions that are smooth in a TV sense, leading to smaller error-bars.

7.3.2 Randomised SVD preconditioning for CG

CG's convergence can greatly be accelerated by using a preconditioner $\mathcal{P}^{-1}$ which approximates $(K_{yy} + b^{-1})^{-1}$. We use randomised SVD methods (Halko et al., 2011; Martinsson and Tropp, 2020), as we find them to be more numerically stable and provide better performance than pivoted Cholesky methods, despite the latter being more common in the literature (Wang et al., 2019). Our preconditioner is based on a randomised approximation of $K_{yy} = \mathcal{T}\,\phi A^{-1}\phi^\top\,\mathcal{T}^\top$ as $\tilde{U}\tilde{\Lambda}\tilde{U}^\top$, where $\tilde{U} \in \mathbb{R}^{c\times r}$ and $r \ll c$. The steps to construct this approximation are

- Constructing a standard normal test matrix $R \in \mathbb{R}^{c\times r}$ with entries sampled from $\mathcal{N}(0, I)$.
- Computing the (thin) QR decomposition $\tilde{Q}\tilde{R} = K_{yy}R$ where $\tilde{Q} \in \mathbb{R}^{c\times r}$ is an orthonormal matrix.
- Constructing symmetric matrix $B \in \mathbb{R}^{r\times r}$ (much smaller than K_{yy}) as $B = Q^T K_{yy} Q$.
- Computing the eigendecomposition $B = V\Lambda V^\top$, and recovering $\tilde{U} = QV$.

This method requires $\mathcal{O}(r)$ matrix vector products with K_{yy} to construct not only an approximate basis but also its complete factorisation. Our approximation is $\mathcal{P} = \tilde{U}\tilde{\Lambda}\tilde{U}^\top + b^{-1}I$ The final step is to invert $\mathcal{P}$ in $\mathcal{O}(r^3)$ time using the Woodbury identity

$$\mathcal{P}^{-1} = bI - b^2\tilde{U}(b\tilde{U}^\top\tilde{U} + \tilde{\Lambda}^{-1})^{-1}\tilde{U}^\top.$$

We choose a value of $r = 400$.

7.3.3 Scalable sample-based hyperparameter learning with the g-prior

As shown in Figure 7.6, stochastic-gradient optimisation of hyperparameters requires thousands of steps. When the number of observations c is large, solving many linear systems to estimate the log determinant gradient at each gradient step becomes too expensive to be practical.

We address this issue by adapting the sample-based EM iteration of Chapter 6 to the 2d CT setting. We use the diagonal g-prior $A = a\,\mathrm{Tr}(\phi^\top\,\mathcal{T}^\top\,\mathcal{T}\,\phi)$, implemented by scaling the feature vectors, as in (6.12). However, unlike in Chapter 6, $n = 1$ and thus computing the scaling vectors in closed form is tractable. For the E-step, we draw samples using the weight-space form of pathwise conditioning, given in (2.31) and re-stated here for the CT

Algorithm 6: Kernelised sampling-based linearised inference for CT

Inputs: Linearised network h, linearisation point $\tilde{v}$, measurements y, discrete Radon transform $\mathcal{T}$, U-Net Jacobian ϕ, initial precision $a > 0$, number of samples m, noise precision $B = bI$

Function `Kvp`$(v,\ a,\ \mathcal{T}\,\phi,\ s,\ \mathrm{B}^{-1})$**:**
  return $\mathcal{T}\,\phi(a^{-1}\mathrm{diag}(s^{\odot 2}))\phi^T\,\mathcal{T}^T\,v + \mathrm{B}^{-1}v$

$s \leftarrow (\sum_{i<c}(\mathcal{T}_i\,\phi)^{\odot 2})^{-1/2}$ `// ` i `indexes output dimensions and` $\odot$ `refers to an operation applied to vector entries elementwise.`

while *a has not converged* **do**
  $\mathcal{P} \leftarrow$ Compute-preconditioner(`Kvp`)
  for $j = 1, \ldots, m$ **do**
    $\zeta_j^0 \leftarrow \mathcal{T}\,\phi(s \odot w_{0,i}) + \mathcal{E}_j$ where $\mathcal{E}_j \sim \mathcal{N}(0, \mathrm{B}^{-1})$ and $w_{0,i} \sim \mathcal{N}(0, A^{-1})$
    $\mathsf{c}_j \leftarrow \mathrm{CG}\left(\mathtt{Kvp}, \zeta_j^0,\ \text{precond.}{=}\mathcal{P}\right)$
    $\zeta_j \leftarrow w_{0,j} - \mathcal{T}\,\phi(a^{-1}\mathrm{diag}(s^{\odot 2}))\phi^T\,\mathcal{T}^T\,\mathsf{c}_j$
  $\delta \leftarrow \mathcal{T}(\phi\tilde{v} - h(\tilde{v}))$
  $\mathsf{c} \leftarrow \mathrm{CG}\,(\mathtt{Kvp},\ y{+}\delta,\ \text{precond.}{=}\mathcal{P})$
  $w_\star \leftarrow s \odot a^{-1}\phi^T\,\mathcal{T}^T\,\mathsf{c}$
  $\hat{\gamma} \leftarrow \frac{1}{m}\sum_{j=1}^{m} \|\,\mathcal{T}\,\phi(s \odot \zeta_j)\|_2^2$
  $a' \leftarrow \hat{\gamma}/\|w_\star\|_2^2$
  $a \leftarrow a'$

Output: Optimised precision a

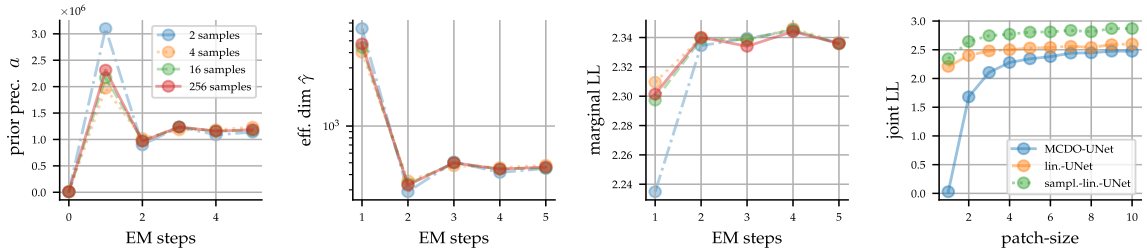


Fig. 7.7 Left 3 plots: traces of prior precision, eff. dim., and marginal test LL vs EM steps for our tomographic reconstruction task with $c = 7680$ described in Section 7.4. Right: joint test LL for varying image patch sizes for sample-based EM inference with the g-prior, inference in the TV-PredCP DIP model (Section 7.2.3, labelled "lin-Unet") and MC Dropout (labelled "MCDO").

setting

$$\zeta_i = w_{0,i} - A^{-1}\phi^\top \mathcal{T}^\top (K_{yy} + b^{-1}I)^{-1} (\mathcal{E}_i + \mathcal{T}\,\phi w_{0,i})$$
$$\text{with} \quad \mathcal{E}_i \sim \mathcal{N}(0, b^{-1}I) \quad \text{and} \quad w_{0,i} \sim \mathcal{N}(0, A^{-1}),$$

and we use preconditioned CG for to solve the linear system. We do not warm-start our CG iteration, drawing new prior and noise samples $(\mathcal{E}_i, w_{0,i})$ at succesive E steps. We find the linear model's posterior mode $w_\star$ by using preconditioned CG to solve (7.11). The preconditioner is described in Section 7.3.2.

There exists unidentifiability between our isotropic noise precision parameter b and the prior precision a. We resolve this by fixing $b = 1$ and setting

$$a = \hat{\gamma}/\|w_\star\|^2 \quad \text{with} \quad \hat{\gamma} = \frac{1}{m}\sum_{i=1}^{m} \|\,\mathcal{T}\,\phi\zeta_i\|_2^2$$

in the M step. We run this EM algorithm, described in detail in algorithm 6, to convergence.

Demonstration: CG sample-based EM iteration

Similarly to image classification (recall Section 6.3), the key hyperparameter is the number of samples to draw for the EM iteration. Again, as shown in Figure 7.7, the number of samples can be kept low (e.g. 2), and we find around 5 steps to suffice for convergence of the prior precision. Our large preconditioner results in CG always hitting the desired low error tolerance within 10 steps. When the problem is small enough for CG to be tractable,

preconditioning makes our kernelised EM algorithm notably faster than its primal form SGD-based counterpart from Section 6.3.

7.3.4 SGD sampling EM iteration for very large reconstructions

In settings where the dimensionality of the observation vector y is very large, i.e. $c \geq 50k$, CG may fail to converge quickly. 3d volumetric reconstruction is an example of such a setting. Here, the the dimensionality of the observation can be larger than the number of parameters of the 3d CNN used for reconstruction, i.e. $c \gtrsim d$. We deal with this, by substituting CG-based sampling for SGD-based sampling in our sample-based EM algorithm, described in Section 7.3.3. That is, we apply algorithm 6, but with every instance of `CG` substituted with `SGD`. We use the Nesterov plus geometric averaging SGD variant for quadratic problems described in Section 4.2.4.

7.3.5 Posterior covariance matrix estimation by sampling

The covariance matrix $K_{x|y}$ is too large to fit into memory for high-resolution tomographic reconstructions. Instead, we draw samples from $\mathcal{N}(x; 0, K_{x|y})$ via pathwise conditioning as

$$x_i = \phi w_{0,i} - \phi A^{-1} \phi^\top \mathcal{T}^\top (K_{yy} + b^{-1} I)^{-1} (\mathcal{E}_i + \mathcal{T}\, \phi w_{0,i})$$
$$\text{with} \quad \mathcal{E}_i \sim \mathcal{N}(0, b^{-1} I) \quad \text{and} \quad w_{0,i} \sim \mathcal{N}(0, A^{-1}).$$

We compute the solution to the linear systems via Preconditioned CG for 2d reconstruction problems and via SGD for 3d problems.

Since only nearby pixels are expected to be correlated, we estimate cross covariances for patches of only up to 10×10 adjacent pixels. Using larger patches yields no improvements. We use the biased, but lower variance, estimator $\hat{K}_{x|y} = \frac{1}{2m}[\sum_{j=1}^{m} \operatorname{diag}(x_j)^{\odot 2} + x_j x_j^\top]$ for $(x_i)_{i=1}^m$ samples from the 0-mean posterior predictive distribution over a given patch.

7.4 Demonstration: uncertainty estimation in CT with the linearised DIP

We now demonstrate the approach on real-measured cone-beam μCT data of a walnut (Der Sarkissian et al., 2019). We first consider reconstructing the middle (2d) slice of the

target volume from sparse measurements in Section 7.4.1. We then demonstrate the scalability of our methods by estimating uncertainty for the full 3d volumetric reconstruction in Section 7.3.4. In all cases, we have access to a "ground truth" reconstruction obtained from an exhaustive dense scan. We use the pre-trained U-net models from Barbano et al. (2022c).In all cases, we have access to a "ground truth" reconstruction obtained from an

7.4.1 Uncertainty estimation for image reconstruction

We begin by reconstructing a 2d image. We target the $501 \times 501\,\text{px}^2$ ($d_x = 251\,001$) central slice of the volumetric data of Der Sarkissian et al. (2019). We consider two levels of sparsity. The first uses a subset of measurements taken from $d_{\mathcal{B}} = 60$ angles and $d_p = 128$ detector rows ($c = 7680$). The second setting uses $d_p = 256$ detector rows and thus c=15360. Here, K is too large to store in memory and K_{yy} too expensive to assemble repeatedly. Our U-net has d=2.97M parameters.

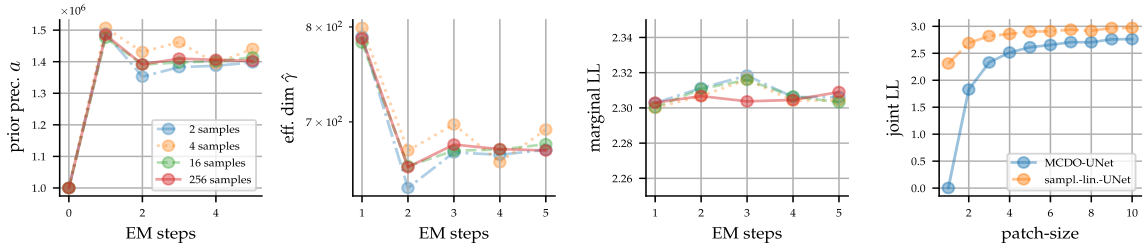


Fig. 7.8 Left 3 plots: traces of prior precision, eff. dim., and marginal test LL vs EM steps for our tomographic reconstruction task with $c = 15360$ described in Section 7.4. Right: joint test LL for varying image patch sizes for sample-based EM inference with the g-prior, inference in the TV-PredCP DIP model (Section 7.2.3, labelled "lin-Unet") and MC Dropout (labelled "MCDO"). In this case, our initialisation for a is close to the optima; its value only changes by around 50% throughout EM iteration, and mostly in the first step.

	$c = 7680$				$c = 15360$			
	LL		wall-clock time (min.)		LL		wall-clock time (min.)	
Method	marginal	(10×10)	params optim.	prediction	marginal	(10×10)	params. optim.	prediction
MCDO-Ug(v)	0.028	2.474	0	$3'$	0.002	2.762	0	$3'$
lin.-Ug(v)	2.214	2.601	$1260'$	$14'$	–	–	–	–
sampl.-lin.-Ug(v)	**2.341**	**2.869**	$12'$	$14'$	**2.310**	**2.972**	$15'$	$14'$

Table 7.1 Tomographic reconstruction: test LL and wall-clock times (A100 GPU) for both 2s reconstruction data sizes.

Comparing hyperparameter learning schemes We consider two different families for the prior precision over the weights A, each is matched with a different inference scheme.

The first is the CNN-block-wise Matérn-$^1/_2$ TV-PredCP prior introduced in Section 7.2.3. We pair it with CG-based marginal likelihood estimation for hyperparameter learning, as described in Section 7.3.1. The large cost of this approach only allows us to perform a single EM step and we are restricted to the smaller $c = 7680$ setting. The second model is the g-prior, which we combine with CG-sampling-based EM iteration, described in Section 7.3.3. We label this method "sampled" in our plots. Unless otherwise specified, we use 16 samples for stochastic EM, and 1024 for prediction. While, the TV-PredCP model's layerwise prior variance and lengthscales take 21 hours to converge (the corresponding optimisation traces are in Figure 7.6, the g-prior model takes only 12 minutes—both on an A100 GPU. These times are provided in Table 7.1. Figure 7.7 and Figure 7.8 show that sample-based EM iteration converges within 4 steps and using as few as 2 samples for both the c=7680 setting and the c=15360 setting (although the reported times use 5 steps and 16 samples). Avoiding explicit estimation of the covariance log-determinant gradient provides us with a two order of magnitude speedup.

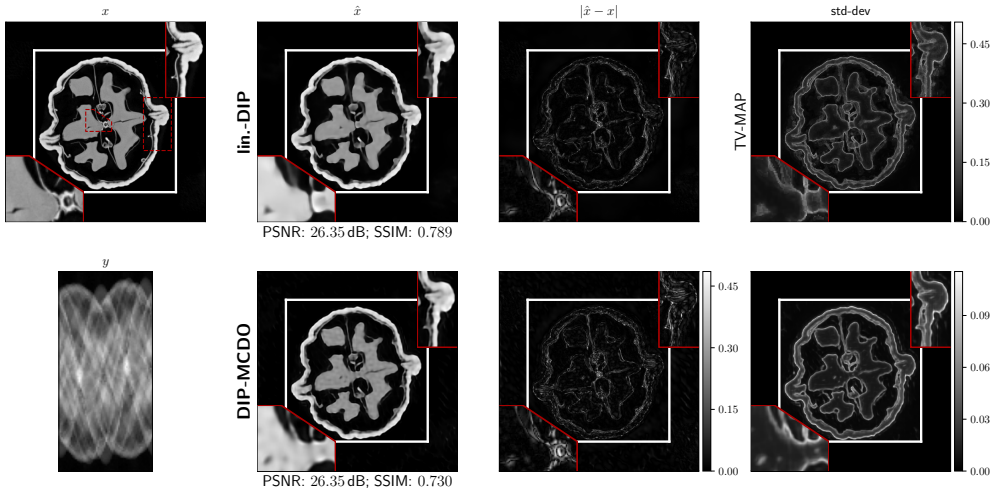


Fig. 7.9 Reconstruction of a $501 \times 501\,\mathrm{px}^2$ slice of a scanned Walnut from $c = 7680$ dimensional measurements using lin.-DIP (using the TV-PredCP prior) and DIP-MCDO along with their respective uncertainty estimates. The zoomed regions (outlined in red) are given in top-left.

Evaluating predictive performance We compare both of our linearised DIP models with MC dropout (MCDO), the most common baseline for NN uncertainty estimation in tomographic reconstruction (Laves et al., 2020; Tölle et al., 2021). Figure 7.9 shows, qualitatively, that the marginal standard deviation assigned to each pixel by the linearised DIP (TV-PredCP) aligns with the pixelwise error in the U-net reconstruction in a fine-grained manner in the $c = 7680$ setting. By contrast, MCDO, spreads uncertainty more uniformly

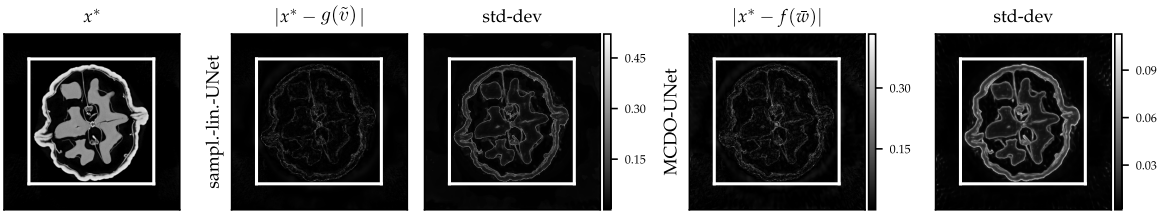


Fig. 7.10 Original 501×501 pixel walnut image and reconstruction error for a c=15360 dimensional observation, along with pixel-wise std-dev obtained with sampling lin. Laplace and MCDO.

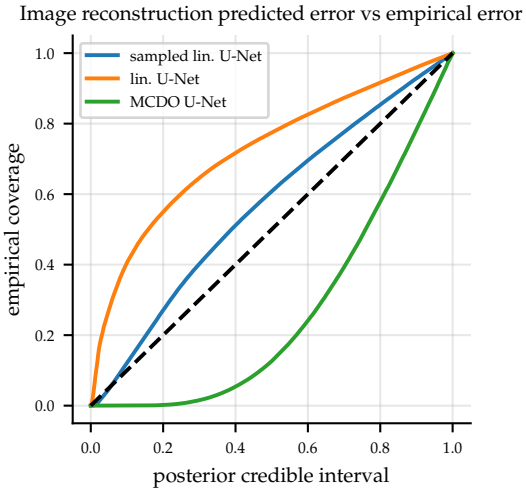


Fig. 7.11 Empirical coverage of test targets for posterior credible intervals of increasing width for our U-net 2d tomographic reconstruction experiment with $c = 7680$. Both linearised DIP variances are under-confident, although the g-prior sampling EM variant is much better calibrated. MCDO is overconfident.

across large sections of the image. Figure 7.10 shows a similar result but for the c=15360 setting and the g-prior DIP model. Table 7.1, Figure 7.7 and Figure 7.8 show that the Log-Likelihood obtained with the g-prior sampling EM DIP exceeds that obtained with the TV-PredCP model, potentially due to the former optimising the prior precision to convergence, while we can only afford a single EM step for the latter. Both methods outperform MCDO, in terms of both marginal and joint LL. Interestingly, MCDO's predictions are very poor marginally but improve significantly when considering covariances.

Uncertainty calibration For the $c = 7680$ setting, we compute normalised residuals by subtracting our predictions from the targets and dividing by the predictive standard deviation. Our predictive distribution for these normalised residuals is the centred unit variance Gaussian. We consider posterior credible intervals centred at 0 and of increasing width and plot the

proportion of test points that fall within them in the left side plot of Figure 7.11. We find dropout inference to underestimate the magnitude of the residuals across all credible interval widths. Linearised inference with TV-PredCP consistently overestimates uncertainty, potentially due to non-converged EM underfitting. The g-prior combined with 5 steps of EM barely overestimates uncertainty, presenting the best overall calibration.

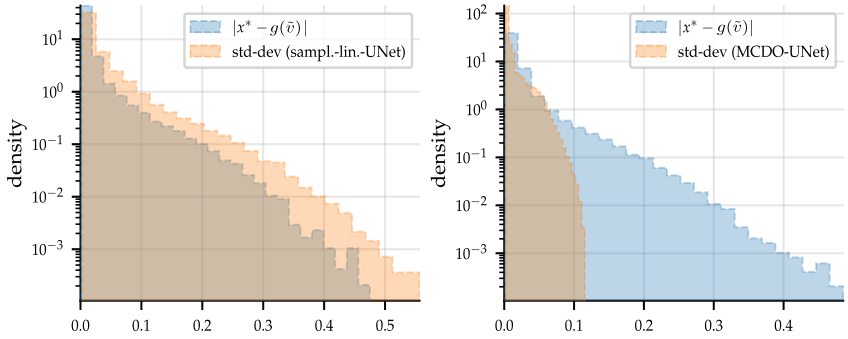


Fig. 7.12 Histogram of the absolute pixelwise error computed between the reconstructed walnut image, given $c = 7680$ observations, and the ground-truth for both lin.-Unet with g-prior (left) and MCDO-Unet (right). We overlay histograms of both methods' predictive standard deviations across pixels.

We further asses calibration by comparing the histogram of the reconstruction errors made by our U-Net to the histogram of marginal, i.e. pixelwise, predictive standard deviation in Figure 7.12 for the $c = 7680$ setting and in Figure 7.13 for the $c = 15360$ setting. In both plots, sample-based linearised Laplace inference slightly overestimates uncertainty and MCDO systematically underestimates uncertainty in the pixels where the reconstruction error is largest. Interestingly, our method shows to be slightly worsely calibrated in the more data-rich setting; the reconstruction error decreases faster than the predictive standard deviation with the addition of new data.

7.4.2 Volumetric uncertainty estimation

We consider 3d reconstruction of the Walnut data with a downscaled resolution of $d_x = (167px)^3 \approx 4,65M$ voxels, from $d_{\mathcal{B}} = 20$ equally distributed angles, and we sub-sample projection rows and columns by a factor of 3. This corresponds to a $c = 1.6M$ dimensional observation space. We use a $d \approx 5M$ parameter 3d CNN. We perform sample-based EM inference, drawing samples with SGD. This procedure is illustrated in Figure 7.15. It is not clear from the plot that EM has converged, but we can not afford the computation for more than 4 steps. To the best of my knowledge, this is the first instance of uncertainty

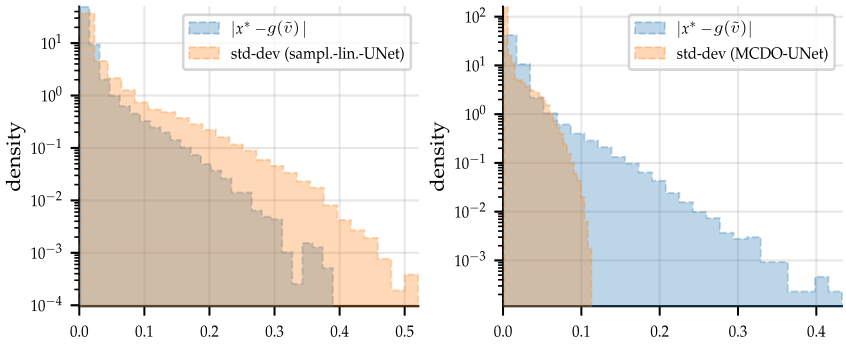


Fig. 7.13 Histogram of the absolute pixelwise error computed between the reconstructed walnut image, given $c = 15360$ observations, and the ground-truth for both lin.-Unet with g-prior (left) and MCDO-Unet (right). We overlay histograms of both methods' predictive standard deviations across pixels.

estimation for deep-learning based volumetric image reconstruction. Three slices of the reconstructed volume, along with their respective error and uncertainty maps are provided in Figure 7.14. We provide error and pixelwise uncertainty histograms in Figure 7.16. Our method underestimates uncertainty in the tails, but this is somewhat alleviated with successive EM steps.

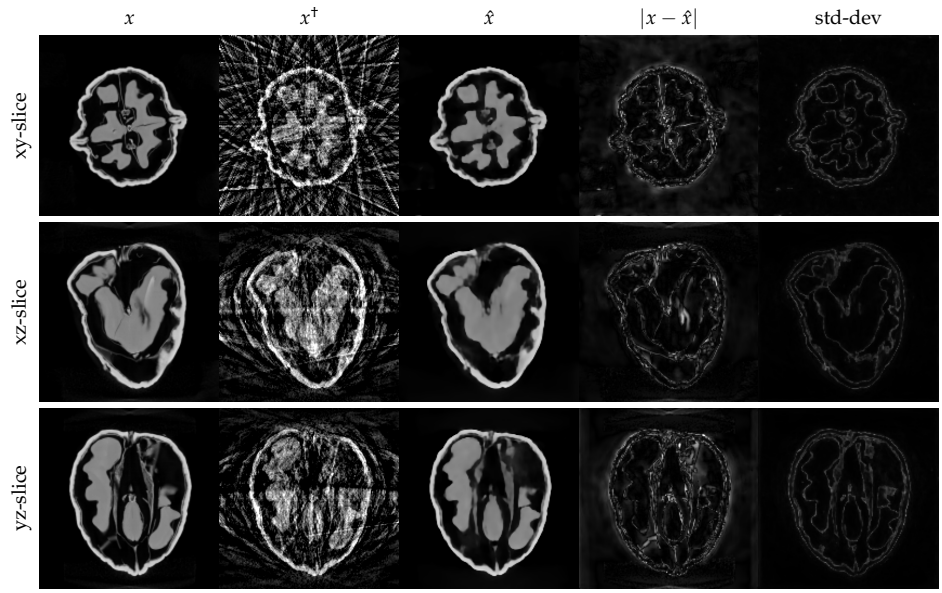


Fig. 7.14 From left to right: 1) Ground truth reconstructions of three $167 \times 167\,\mathrm{px}^2$ slices from the $167 \times 167 \times 167\,\mathrm{px}^3$ Walnut data from Der Sarkissian et al. (2019). 2) Filtered backprojections (i.e. reconstructions obtained by pseudoinverting the operator $\mathcal{T}$) from $c = 1.6M$ observations. 3) Unet reconstructions. 4) Absolute error in Unet reconstructions. 5) Pixelwise standard deviations obtained with the linerised Unet and the g-prior.

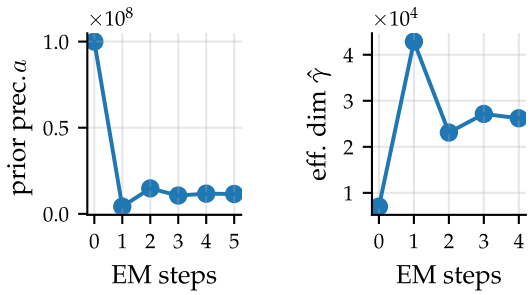


Fig. 7.15 Traces of prior precision α and eff. dim. $\hat{\gamma}$ vs EM steps for the $c = 1.6M$ 3d volumetric reconstruction task.

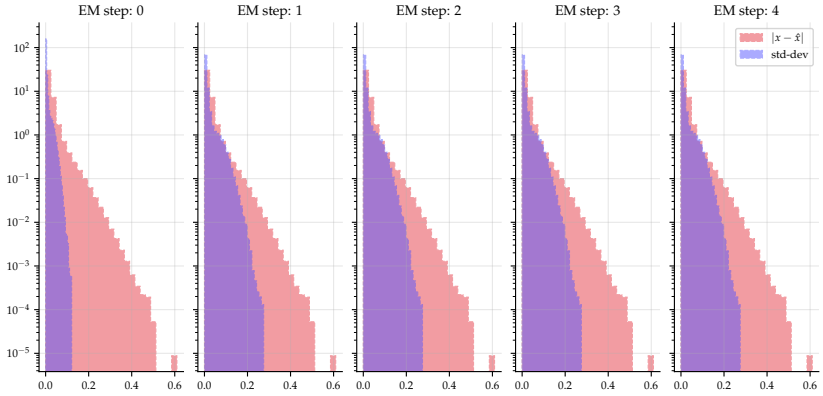


Fig. 7.16 Histograms (y-axes are normalised to represent empirical densities) of the voxel-wise error computed between the reconstructed 3d volumetric walnut and the ground-truth, along with the histograms of pixelwise predictive standard deviations across voxels.

7.5 Linearised DIP Bayesian experimental design for CT

In CT, Bayesian experimental design leverages an a-priori model to select the scanning angles which are expected to yield the highest fidelity reconstruction. Adaptive design further incorporates information gained at previous angles to inform subsequent angle selections (Chaloner and Verdinelli, 1995). These methods are of great practical interest since they promise to reduce radiation dosages and scanning times. Alas, existing CT design methods often struggle to improve over equidistant angle choice (Shen et al., 2022). Furthermore, the requisite of additional computations before subsequent scans makes adaptive methods impractical for many applications.

Critically important to experimental design is the choice of prior (Feng, 2015; Foster, 2021). Linear models allow for tractable computation of quantities of interest for experimental design, but their predictive uncertainty is independent of previously measured values, disallowing adaptive design (Burger et al., 2021). More complex model choices make inference difficult, necessitating approximations which can degrade performance (Helin et al., 2022b; Shen et al., 2022).

This section aims to make adaptive design practical by considering a setting where the CT scan is performed in two phases. First, a sparse pilot scan is performed to provide data with which to fit a adaptive methods. These are then used to select angles for a full scan. We demonstrate this procedure with a synthetic dataset where a different "preferential" angle is most informative for each image. Preferential directions appear commonly in industrial CT for material science and in medical CT for medical implant assessment. We use the linearised Deep Image Prior (DIP) (Barbano et al., 2022a) as a data-dependent prior for adaptive design which preserves the tractability of conjugate Gaussian-linear models. Unlike simple linear models, the linearised DIP outperforms the equidistant angle baseline. Finally, we show that designs obtained with the linearised DIP perform well under traditional (non DIP-based) regularised-reconstruction.

Section 7.5.1 covers sequential inference in the conjugate Gaussian-linear setting. Section 7.5.2 introduces experimental design with linear models and linearised neural networks. Finally, Section 7.6 demonstrates our approach on a synthetic CT scanning angle selection task.

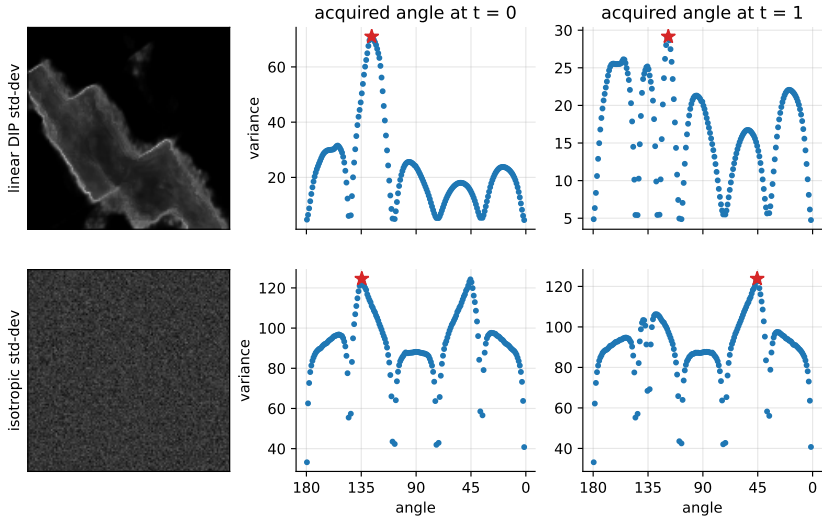


Fig. 7.17 Top row: the linearised DIP assigns prior variance to pixels where edges are present, guiding angle selection so that X-ray quanta cover these pixels. Bottom row: the isotropic linear model's variance does not depend on the measurements. Angles 45 and 135 are chosen since they are oblique and maximise quanta path-length in the image.

7.5.1 Sequential inference with linear(ised) models

Let $\mathcal{B}_a$ be the set of *all* possible angles at which we can scan. The task is to choose the subset of angles $\mathcal{B} \subset \mathcal{B}_a$ which produces the highest-fidelity reconstruction. We shall add angles sequentially over T steps. The set $\mathcal{B}^{(t)}$ denotes the chosen angles up to step $t < T$, and $\bar{\mathcal{B}}^{(t)} = \mathcal{B}_a \setminus \mathcal{B}^{(t)}$ the angles left to choose from. $\mathcal{B}^{(0)}$ denotes the set of angles used in the initial pilot scan, and $\mathcal{B} = \mathcal{B}^{(T)}$ the full design. We incorporate a decision to scan at angle $\beta \in \bar{\mathcal{B}}^{(t)}$ by concatenating the matrix $\mathcal{T}^{\beta} \in \mathbb{R}^{d_p \times d_x}$, which contains a row for each detector pixel at angle β, to the operator. After step t, the operator $\mathcal{T}^{(t)} \in \mathbb{R}^{d_p \cdot d_{\mathcal{B}^{(t)}} \times d_x}$ stacks $d_{\mathcal{B}^{(t)}}$ of these matrices, with $d_{\mathcal{B}^{(t)}} = |\mathcal{B}^{(t)}|$. $\bar{\mathcal{T}}^{(t)} \in \mathbb{R}^{d_p \cdot d_{\bar{\mathcal{B}}^{(t)}} \times d_x}$ denotes the forward operator for the angles left to choose from.

For design, we place a multivariate Gaussian prior on x with zero mean and covariance matrix $K \in \mathbb{R}^{d_x \times d_x}$. Together with the Gaussian noise model in (7.1), this gives a conjugate Gaussian-linear model. The vector $y^{(t)} \in \mathbb{R}^{d_p \cdot d_{\mathcal{B}^{(t)}}}$ of all measurements at step t is distributed as

$$y^{(t)}|x \sim \mathcal{N}(\mathcal{T}^{(t)}\, x,\, b^{-1} I_c) \quad \text{with} \quad x \sim \mathcal{N}(0, K).$$

Thus, $K_{yy}^{(t)} + b^{-1}I$, with $K_{yy}^{(t)} = \mathcal{T}^{(t)} K (\mathcal{T}^{(t)})^\top$, is the measurement covariance and the posterior over x is

$$\begin{aligned} x|y^{(t)} &\sim \mathcal{N}(\mu_{x|y^{(t)}}, K_{x|y^{(t)}}), \\ \text{with} \quad \mu_{x|y^{(t)}} &= K(\mathcal{T}^{(t)})^\top (K_{yy}^{(t)} + b^{-1}I)^{-1} y^{(t)}, \\ \text{and} \quad K_{x|y^{(t)}} &= K - K(\mathcal{T}^{(t)})^\top (K_{yy}^{(t)} + b^{-1}I)^{-1} \mathcal{T}^{(t)} K. \end{aligned} \tag{7.14}$$

The predictive covariance $K_{x|y^{(t)}}$ completely characterises the uncertainty of the reconstruction at step t and is the building block for the angle selection criteria in Section 7.5.2.

With this, a concern may be that natural images often exhibit heavy-tailed non-Gaussian statistics (Seeger and Nickisch, 2011). Furthermore, by (7.14), $K_{x|y^{(t)}}$ depends on the choice of angles through $\mathcal{T}^{(t)}$, but not on the measurements made at said angles $y^{(t)}$, precluding adaptive design. In Section 7.5.3, we will address both of these concerns by constructing a very flexible data dependent covariance kernel from the Jacobian of a NN, recovering adaptive design capabilities.

7.5.2 Experimental design with linear(ised) models

Acquisition objectives. Since the linear design task is submodular (Seeger, 2009), we greedily add one single angle per acquisition step [3]. We consider two popular acquisition objectives.

The first objective, *expected information gain* (EIG) (Mackay, 1992b), is the expected reduction in the posterior entropy $\mathbb{H}(\mathcal{P}_{x|y})$ from scanning at angle β. At step t, it is given by

$$\text{EIG} := \mathbb{H}(\mathcal{P}_{x|y^{(t)}}) - \mathbb{E}_{y^\beta|y^{(t)}}[\mathbb{H}(\mathcal{P}_{x|y^{(t)},y^\beta})] = \log\det(b^{-1} I_{d_{\mathcal{B}^{(t)}}} + \mathcal{T}^\beta K_{x|y^{(t)}} (\mathcal{T}^\beta)^\top) + C \tag{7.15}$$

where the constant $C = -\log\det(b^{-1}I)$ is independent of the angle choice. Intuitively, the determinant of the matrix $\mathcal{T}^\beta K_{x|y^{(t)}} (\mathcal{T}^\beta)^\top \in \mathbb{R}^{d_p \times d_p}$ penalises angles for which different detector elements make correlated measurements and the log term encourages the measurements from all detector pixels to be similarly informative. EIG is known as a (D)eterminant-optimal objective.

[3] Submodularity guarantees this procedure obtains a score within a $(1 - 1/e)$ factor of the optimal strategy.

Derivation Expected information gain

The entropy of a multivariate Gaussian is $\mathbb{H}(\mathcal{N}(\mu, K)) = \frac{1}{2}\log\det(K) + \frac{d}{2}(\log(2\pi)+1)$. We compute the posterior covariance log-determinant at time t from the covariance at time $t-1$ using the matrix determinant lemma

$$\begin{aligned}\log\det(K_{x|y^{(t)}}) = &-\log\det(K^{-1}_{x|y^{(t-1)}}) - \log\det(bI) \\ &- \log\det(b^{-1}I + \mathcal{T}^{(t)}\, K_{x|y^{(t-1)}}\, \mathcal{T}^{\top,(t)}).\end{aligned}$$

Note that both sides of the equality are independent of the targets y. Thus we drop the expectation in (7.15). With that, we have

$$\begin{aligned}\text{EIG} &= \log\det(K_{x|y^{(t-1)}}) - \log\det(K_{x|y^{(t)}}) \\ &= \log\det(K_{x|y^{(t-1)}}) - [-\log\det(K^{-1}_{x|y^{(t-1)}}) - \log\det(bI) \\ &\quad - \log\det(b^{-1}I + \mathcal{T}^{\beta}\, K_{x|y^{(t-1)}}(\mathcal{T}^{\beta})^{\top})] \\ &= -\log\det(b^{-1}I) + \log\det(b^{-1}I + \mathcal{T}^{\beta}\, K_{x|y^{(t-1)}}(\mathcal{T}^{\beta})^{\top}) \\ &= \log\det(b^{-1}I + \mathcal{T}^{\beta}\, K_{x|y^{(t-1)}}(\mathcal{T}^{\beta})^{\top}) + C\end{aligned}$$

where the constant $C = -\log\det(b^{-1}I)$ is independent of angle choice, yielding the angle selection objective.

Remark What information are we gaining?
EIG quantifies the information (in nats) we expect to gain by observing the detector elements' measurements for an angle or set of angles (Mackay, 1992b). EIG is also equal to the mutual information between the reconstruction x and the new measurement y^{β} conditional on the previous measurements $y^{(t-1)}$, i.e. $MI(x, y^{\beta}|y^{(t-1)})$, giving an interpretation as aiming to select the angle β most informative towards the reconstruction. For fixed model hyperparameters, EIG is always greater or equal than 0 since making additional measurements cannot increase the uncertainty in the reconstruction.

The second objective, which we find to perform better empirically, is to choose the angles for which our prediction has the largest *expected squared error* (ESE) in measurement space

$$\text{ESE} := \mathbb{E}_{y^{\beta},\, x|y^{(t)}}[(y^{\beta} - \mathcal{T}^{\beta}\, x)^{\top}(y^{\beta} - \mathcal{T}^{\beta}\, x)] = \text{Tr}(\mathcal{T}^{\beta}\, K_{x|y^{(t)}}(\mathcal{T}^{\beta})^{\top}) + C. \tag{7.16}$$

This objective is equivalent to EIG in the setting where our detector has a single pixel.

Remark Motivating ESE
The ESE objective in (7.16) aims to minimise the squared prediction error in measurement space. Objectives of this kind are commonly known as (A)verage-optimal. However, ESE is A-optimal over measurement space y, not over image space x. ESE is crucially different from minimising the arguably more relevant expected squared reconstruction error, a more computationally expensive criterion. ESE can be understood as a naïve simplification of EIG, by discarding correlations between detector pixels, making $\log\det(\mathcal{T}^\beta\, K_{x|y^{(t-1)}}(\mathcal{T}^\beta)^\top)$ match $\sum_{i\leq d_p}\log[\mathcal{T}^\beta\, K_{x|y^{(t-1)}}(\mathcal{T}^\beta)^\top]_{ii}$. Then, the order of log and sum are switched, something that will only preserve the output (up to a constant independent of β) if every element under the sum is the same. Having reached this point, since the log function is monotonic, it does not affect angle selection and the criterion matches the trace of $\mathcal{T}^\beta\, K_{x|y^{(t-1)}}(\mathcal{T}^\beta)^\top$.

Efficient acquisition. Constructing the matrix $\mathcal{T}^\beta\, K_{x|y^{(t)}}(\mathcal{T}^\beta)^\top$ repeatedly for each candidate angle $\beta \in \bar{\mathcal{B}}^{(t)}$ requires $\mathcal{O}(d_p \cdot d_{\bar{\mathcal{B}}^{(t)}})$ matrix vector products, which is very costly even for moderate size scanners. Instead, we estimate the matrix for every angle simultaneously by drawing m samples from $\mathcal{N}(0, \bar{\mathcal{T}}^{(t)} K_{x|y^{(t)}}(\bar{\mathcal{T}}^{(t)})^\top)$. That is, we sample $\mathbb{R}^{d_p \cdot d_{\bar{\mathcal{B}}^{(t)}}}$ sized vectors containing the concatenated "pseudo measurements" for each unused angle $\beta \in \bar{\mathcal{B}}^{(t)}$. We again use pathwise conditioning

$$\bigoplus_{\beta\in\bar{\mathcal{B}}^{(t)}} y_i^\beta = \bar{\mathcal{T}}^{(t)}\Big(x_i - K(\mathcal{T}^{(t)})^\top(K_{yy}^{(t)} + b^{-1}I)^{-1}(\mathcal{E}i + \mathcal{T}^{(t)}\, x_i)\Big) \quad \text{with}$$
$$x_i \sim \mathcal{N}(0, K) \quad \text{and} \quad \mathcal{E}_i \sim \mathcal{N}(0, b^{-1}I). \tag{7.17}$$

Here, $i \in \{1, ..., m\}$ indexes different samples and $\bigoplus$ denotes the concatenation of vectors generated for each $\beta \in \bar{\mathcal{B}}^{(t)}$. Now, for each angle $\beta \in \bar{\mathcal{B}}^{(t)}$, we compute

$$\mathcal{T}^\beta\, K_{x|y^{(t)}}(\mathcal{T}^\beta)^\top \approx \frac{1}{m}\sum_{i=1}^m y_i^\beta (y_i^\beta)^\top,$$

which is then used to estimate the acquisition objective (7.15) or (7.16). The log term makes EIG estimates only asymptotically unbiased (i.e. as $m \to \infty$) but we find the bias to be insignificant. Once the angle β that maximises (7.15) or (7.16) is chosen, we update $K_{yy}^{(t+1)}$ as

$$K_{yy}^{(t+1)} = \begin{bmatrix} K_{yy}^{(t)} & \mathcal{T}^{(t)}\, K(\mathcal{T}^{(t+1)})^\top \\ \mathcal{T}^{(t+1)}\, K(\mathcal{T}^{(t)})^\top & \mathcal{T}^{(t+1)}\, K(\mathcal{T}^{(t+1)})^\top \end{bmatrix}, \tag{7.18}$$

and repeat the procedure, i.e. return to (7.17).

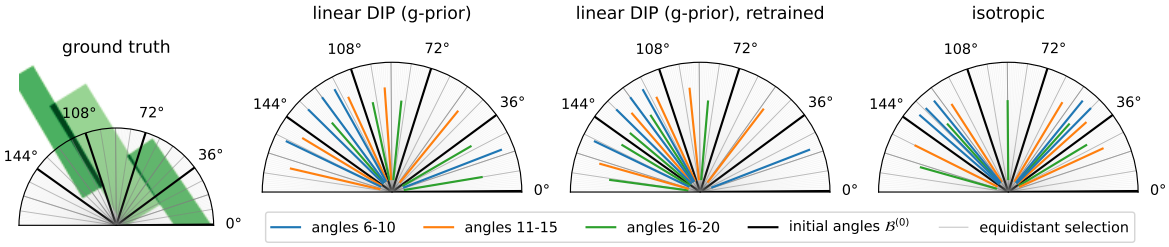


Fig. 7.18 First 20 angles selected by each method under consideration for an example image.

7.5.3 Construction of the prior covariance K

Now we describe the construction of the Gaussian prior covariance $K \in \mathbb{R}^{d_x \times d_x}$ over reconstructions. We consider a range of models, building from very simple models to flexible data-driven ones that allows for adaptive design.

Isotropic model. The simple choice $K = I_{d_x}$ assumes uncorrelated pixels, and it implies a ridge regulariser for the reconstruction, which is known to perform poorly in imaging.

Matérn-$1/2$ Process. Antoran et al. (2023), and also Section 7.2.3, employ the Matérn-$1/2$ covariance $[K]_{ij,i'j'} = \exp(-\psi^{-1}\sqrt{(i-i')^2 + (j-j')^2})$, where (i, j) index the pixel locations in the image x in terms of height and width respectively, as a surrogate for the TV regulariser.

Linearised deep image prior This data-driven prior is constructed by first fitting a DIP model on the measurements taken during the pilot scan with (7.4). We then adopt a linear model on the basis expansion given by the Jacobian of the trained U-net, denoted $\phi \in \mathbb{R}^{d_x \times d}$. The resulting covariance matrix $K = \phi A^{-1} \phi^\top$ incorporates information about the pilot measurements on which the NN was trained through its Jacobians ϕ. It assigns higher prior variance being near the edges in the reconstruction (this is shown in Figure 7.17), which are most sensitive to a change in U-net parameters. The covariance $A^{-1} \in \mathbb{R}^{d \times d}$ weights different Jacobian entries. We consider two different structures for A^{-1}.

- The filter-wise block-diagonal matrix of Section 7.2.3. This choice uses a large number of hyperparameters and thus risks overfitting to the pilot scan measurements.
- The neural g-prior, introduced in Section 5.3.2. We implement it through feature scaling, as described in Section 6.2.2. We update the feature scaling vectors every 5 acquired angles.

The Matérn model has its lengthscale as a free hyperparameter. Learning this hyperparameter from the data makes the model adaptive. The filter-wise DIP prior has filter-wise marginal variances and lengthscales. We set these such that the model evidence is maximised given the pilot scan measurements using gradient-based optimisation. Since the number of pilot observations is small, the exact evidence (2.43) is tractable. We omit the global prior variance scaling hyperparameter from all models since the choice of this value only alters the width of the posterior errorbars, but not their shape. As a result, experimental design is invariant to the choice of global prior variance scaling[4]. The same is true for the isotropic observation noise precision b.

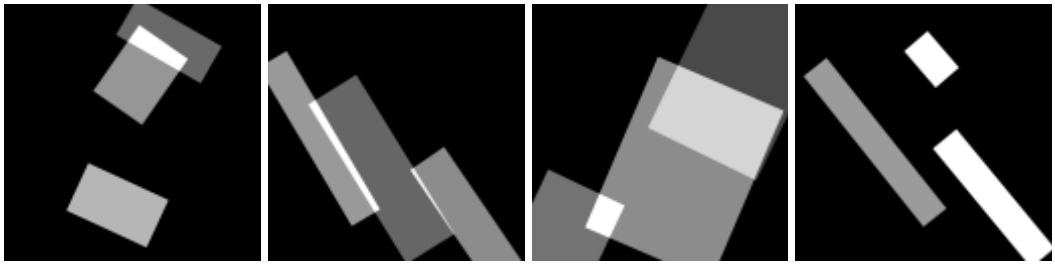


Fig. 7.19 Examples of synthetic images used for our experiments.

7.6 Demonstration: designing CT angle selection strategies

We now demonstrate the experimental design objectives from Section 7.5.2 coupled with the models from Section 7.5.3. In almost all real-world CT deployments, the scanning angles are chosen to be equidistant. This strategy is known to be very hard to beat, and we will use it as our strong baseline. We will also test weather the DIP-based designs work well exclusively for DIP-based reconstructions or if they generalise to non-NN-based reconstruction methods.

Experimental setup We simulate CT measurements y from 128×128 ($d_x = 16384$) pixel images of 3 superimposed rectangles. Their orientation is sampled from a single normal distribution with zero mean and standard deviation $2.86°$. Thus, images in this class contain edges in roughly two perpendicular "preferential" directions (see Figure 7.18 and Figure 7.19). We simulate CT measurements by applying the discrete Radon transform operator $\mathcal{T} \in \mathbb{R}^{c \times d_x}$ and adding Gaussian noise with standard deviation matching 5 % of the average absolute value of the noiseless measurements $\mathcal{T}\, x$ to generate y. We divide the scanning range $[0°, 180°)$ into 200 selectable angles (i.e. $|\mathcal{B}_a| = 200$). The pilot scan measures at 5

[4]I was made aware of this by David Janz via personal communication and then I verified it experimentally.

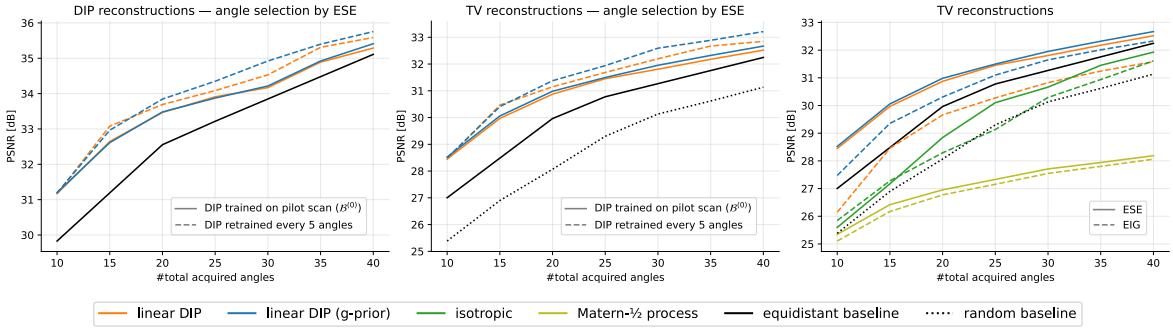


Fig. 7.20 Reconstruction PSNR vs n. angles scanned, averaged across 30 images (5% noise).

equidistant angles, on which we fit all models' hyperparameters and the linearised DIP's U-net. Then, we apply the methods in Section 7.5.2 to produce designs consisting of 35 additional angles. For every 5 acquired angles, we evaluate reconstruction quality using both the DIP (7.4), and the traditional NN-free TV regularised approach (7.3). We include equidistant and random angle selection as strong and weak baselines, respectively. On an A100 GPU, a full linearised DIP acquisition step with $K = 3000$ samples takes 9 seconds and the full design takes 5 minutes.

For the **linearised DIP**, we consider both training our U-net and prior hyperparameters only on the pilot scan, and also retraining every 5 angles. Figure 7.18 shows both approaches can identify and prioritise the preferential direction, leading to reconstructions that *outperform the equidistant angle baseline by over 1.5 dB* in the range of $[10, 15]$ angles. This is shown in Figure 7.20. During this initial stage, the linearised DIP requires roughly *30% less scanned angles* to match the equidistant baseline's performance. The performance gap decreases as we select more angles, although linearised DIP remains more efficient even after 40 angles. Retraining the U-net provides most benefits in the large angle regime. It increases focus on preferential directions and consistently provides gains >0.5dB after 20 angles. All gains over the equidistant baseline are obtained with both DIP reconstruction (7.4) and traditional TV regularised reconstruction (7.3).

The **isotropic and Matérn-$1/2$** models' uncertainty estimates are independent of the pilot measurements. These models prioritise clustered sets of oblique angles which maximise the length of quanta trajectories in the image. They perform similar to or worse than random. This negative result is due to the lengthscale hyperparameter overfitting to the small amount of data from the pilot scan and taking very large values. This makes the predictive variance insensitive to previous acquisitions, as shown in Figure 7.21. For contrast, we display the g-prior DIP's acquisitions in Figure 7.22.

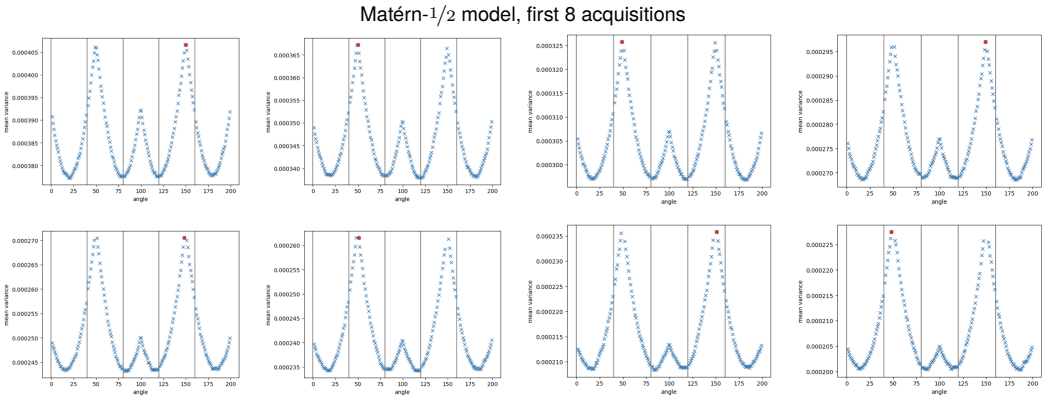


Fig. 7.21 Variance assigned to each candidate angle during the first 8 design steps by our Matérn-$1/2$ model.

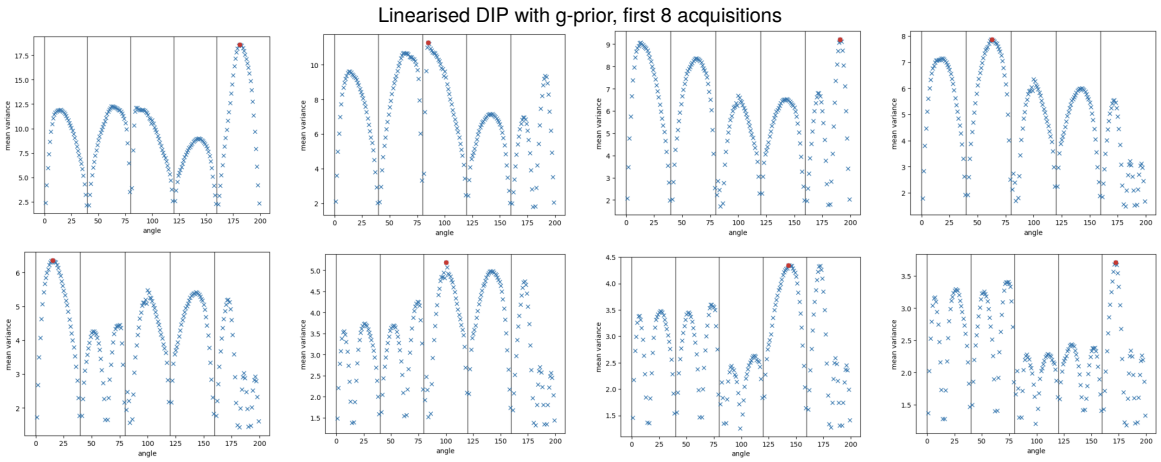


Fig. 7.22 Variance assigned to each candidate angle during the first 8 design steps by our linearised DIP model with the g-prior.

ESE outperforms EIG across models. For the linearised DIP, this gap is smaller when using the g-prior. This is surprising, given that EIG takes covariances into account, but ESE doesn't. We hypothesise that model misspecification and hyperparameter overfitting may result in poor measurement covariance estimates, in turn degrading the EIG estimates.

Remark On the dangers of combining model evidence maximisation with data acquisition

Figure 7.21 shows the Matérn-$1/2$ model concentrates its selection on oblique angles, and this does not change as more data is acquired. This results in a very non-diverse angle set which achieves very poor performance. To understand why this happens

we first remark that the Matérn-$1/2$ model generalises the isotropic model and the two are equal when the lengthscale is set to $\psi = 0$. We investigate the hyperparameters chosen by the model evidence for the Matérn-$1/2$ model and find that for all images the lengthscale is in the range [40-70]. This value is very large relative to the size of the image (128×128) and represents an assumption that the reconstructed image has only 2 or 3 regions with different pixel intensity values. Under this assumption, only taking measurements at 3 different angles is justified. Each new angle introduced into the operator reduces the predictive variance of every unseen angle almost equally. As a result, the relative assignment of predictive variance in angle space remains roughly constant throughout design steps.
Although it is well known that experimental design is very sensitive to the choice of prior (Feng, 2015; Foster, 2021), the ease with which the relatively simple Matérn-$1/2$ model can overfit was unexpected to us.

7.7 Discussion

Having laid the groundwork for scalable uncertainty estimation and hyperparameter selection for linearised neural networks in Chapter 4, Chapter 5 and Chapter 6, this chapter has applied these advances to tomographic reconstruction. In particular, we have introduced a probabilistic formulation of the deep image prior (DIP) that utilises a linearisation of the network around the parameters that output the candidate reconstruction. The approach yields far better uncertainty estimates on 2d image reconstructions from real-measured μCT data than MC dropout-based approaches standard in the field of CT. Furthermore, our method is the first to have been applied to 3d volume reconstructions.

Motivated by standard practise in the field of CT, we developed a bespoke TV-based prior for our linearised NN. However, we found it to be dominated, in terms of both computational efficiency and calibration of uncertainty estimates, by the more general diagonal g-prior, introduced in Section 5.3.2.

Finally, we applied linearised DIP inference to adaptively select scanning angles for CT. Our results suggest that dependence on the measurement data, i.e. adaptivity, is key to outperforming equidistant angle selection, a notoriously strong baseline in CT reconstruction (Helin et al., 2022b; Shen et al., 2022). Distinctly from previous work, our methods only necessitate a pilot scan instead of being fully online, increasing applicability. We observe the largest gains in the 10 to 20 angle regime, where our designs reduce the angle requirement

by roughly 30% without loss of reconstruction quality. This is true for both traditional TV-regularised and DIP-based reconstructions.

With this, we conclude the technical content of this thesis. The following chapter reviews the main contributions of the work and discusses exciting avenues for future work.

Chapter 8

Conclusions and future work

This thesis has studied the problem of large scale Bayesian inference in linear models and neural networks. We made contributions of both fundamental nature, furthering our understanding of linearised neural networks, and also of practical nature by introducing a number of learning algorithms that scale well in both the number of observations and model parameters. We have strived for these methods to be fully compatible with existing (and hopefully future) progress in the field of deep learning. I hope that this work will contribute towards the development and real-world deployment of uncertainty-aware data-driven decision making systems.

We go on to provide a recap of our contributions in Section 8.1, while giving a critical overview of their strengths and weaknesses. In Section 8.2 we discuss avenues for future work.

8.1 Recap of contributions

Chapter 4 proposed using SGD, the workhorse algorithm of deep learning, to perform posterior inference in Gaussian processes. Traditionally, lack of scalability has been a major impediment to the use of these models; the cost of exact inference is cubic in the number of observations, i.e. $\mathcal{O}(n^3)$. SGD has not been considered for this task in the past, in part because it provides worse formal convergence guarantees than alternatives like CG (Boyd and Vandenberghe, 2014). At a high level, our key insight is that full convergence of SGD is not necessary to obtain good performance. SGD converges very fast in the directions of parameter space that matter for prediction, and very slowly in others. Additionally, this

convergence is monotonic in the number of steps, making SGD an anytime method amenable to early stopping. These two features, combined with SGD's linear cost, i.e. $\mathcal{O}(n)$, per step allows it to handily outperform other inference schemes when dealing with datasets of more than $n \approx 100$k observations. The weakness of SGD is its lack of convergence in most low-eigenvalue directions of parameter space. These make little, but some, difference for prediction, resulting in SGD always providing an approximate solution. Thus, below $n \approx 100$k, conjugate gradients is likely to converge faster while providing an effectively exact solution.

Chapter 5 studied the applicability of the linearised Laplace model evidence to modern neural networks. Motivated by the heavy dependence on this parameter of the calibration of the linearised Laplace errorbars, we focused on learning the prior precision with the evidence. We first interrogated the validity of the Laplace approximation's assumption that we Taylor expand about a mode of the posterior. In fact, satisfying this assumption is not practical in modern deep learning. Nevertheless, we showed that every expansion point implies an associated basis function linear model. As we use this model to provide errobars, we propose to choose hyper- parameters using the evidence of this model. This requires only the solving of a convex optimisation problem, one much simpler than NN optimisation. We then showed that, for neural networks with normalisation layers—that is, practically all modern architectures—the predictive posterior covariance can only be identified up to a scalar constant, or a constant per normalised group of weights. We introduce two prior classes which produce a predictive posterior invariant to this scaling constant. The first is a diagonal Gaussian prior with layerwise precision parameters fit to maximise the model evidence. The second is the diagonal g-prior, which only has an isotropic scaling parameter that can be set to any value.

Chapter 6 combines the efficient SGD posterior sampling from Chapter 4 with the developments in linearised model hyperparameter learning of Chapter 5 into a scalable sample-based EM algorithm. The key component is our M step, which builds upon MacKay (1992a)'s update for the prior precision. Our update makes much more progress per step than traditional gradient based optimisation with the model evidence and can be estimated with only posterior samples. We combine these methods with a number of matrix-free linear algebra techniques and SGD warm starting to scale linearised inference to ResNet-50 (25M parameters) and Imagenet (1.2M observations and 1000 output dimensions). To the best of our knowledge, this is the first time Bayesian inference has been performed in this setting without assuming some degree of independence across weights in the model. Linearised inference performs particularly well in terms of joint predictions, which are key to sequential decision making. However, despite our methods being more accurate and scalable than other

Bayesian approximations, they still introduce very significant overhead compared to training a single neural network. Furthermore, we used the diagonal g-prior in our experiments. Since this prior only has a single free parameter, it may be cheaper to set its value with cross validation than to use our EM iteration.

Chapter 7 applies the methods developed in chapters 4 to 6 to uncertainty estimation in CT reconstructions from the deep image prior. On 2d images, we obtain more calibrated uncertainty estimates than previous probabilistic approaches to DIP reconstruction. The scalability of our inference methods allows us to apply them to uncertainty estimation in 3d volumetric reconstructions from the deep image prior. To the best of our knowledge, this is the first time neural network uncertainty has been estimated on this large scale task. We concluded by leveraging the errorbars from the linearised deep image prior to guide scanning angle selection in CT. This allows us to reduce the number of scans needed to obtain a constant reconstruction quality. We also constructed a bespoke total-variation based prior for the linearised DIP, but we found its performance dominated by the more-scalable diagonal g-prior. This was true for both uncertainty estimation and experimental design. Perhaps we should have payed more attention to the Richard Sutton quote that preceded Chapter 6.

8.2 Future Work

Scalable hyperparameter learning for GPs and linearised neural networks A clear avenue for future work is leveraging SGD posterior sampling to learn GP hyperparameters. One way to do this is to use these posterior samples in the existing Hutchinson estimator of the evidence log-determinant gradient. However, I am not optimistic about this direction because gradient-based optimisation of the evidence requires many steps. This makes updating our samples for each new step is very expensive. It would be more interesting to generalise the MacKay update, used in Chapter 6, beyond marginal prior precisions.

Online Laplace for normalised networks The developments of Chapter 5 are focused on the post-hoc setting, where we have access to a pre-trained neural network. An exciting line of research is online Laplace, where the hyperparameters are learnt simultaneously with the network weights. However, these methods are incompatible with normalisation layers, ostensibly for the same reasons described in Chapter 5. In Lin et al. (2023a), a paper not included in this thesis, we did some work relating online Laplace methods to the tangent linear model. It would be good to further leverage this connection, and the results of Chapter 5, to make online Laplace amenable to normalisation layers.

Sequential decision making with neural networks It seems plausible that given large enough datasets, modern large-scale neural models will rarely encounter out of distribution scenarios. Thus, the utility of model uncertainty as a tool for rejecting spurious model behaviour may decrease. However, I do not think that the more general problem of sequential decision making can be solved in the same way. Thus, I am particularly optimistic about this application of of Bayesian inference with neural networks. In particular, I am excited about the use of the linearised DIP to design CT scanning strategies for the real-world. Furthermore, the experimental design methods of Chapter 7 may be applied to magnetic resonance imaging, where the forward operator is a Fourier transform, almost out of the box.

References

Adam, V., Chang, P. E., Khan, M. E., and Solin, A. (2021). Dual parameterization of sparse variational Gaussian processes. In *Advances in Neural Information Processing Systems 34, NeurIPS*.

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*.

Allingham, J. U., Antorán, J., Padhy, S., Nalisnick, E., and Hernández-Lobato, J. M. (2022). Learning generative models with invariance to symmetries. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*.

Amari, S., Park, H., and Fukumizu, K. (2000). Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Comput.*

Andrei, N. (2009). Accelerated conjugate gradient algorithm with finite difference hessian/vector product approximation for unconstrained optimization. *Journal of Computational and Applied Mathematics*.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine Learning*.

Antorán, J. (2019). *Understanding Uncertainty in Bayesian Neural Networks*. PhD thesis, University of Cambridge.

Antorán, J., Allingham, J., and Hernández-Lobato, J. M. (2020). Depth uncertainty in neural networks. *Advances in Neural Information Processing Systems 33, NeurIPS*.

Antorán, J., Allingham, J., Janz, D., Daxberger, E., Nalisnick, E., and Hernández-Lobato, J. M. (2022). Linearised Laplace inference in networks with normalisation layers and the neural g-prior.

Antorán, J., Allingham, J. U., and Hernández-Lobato, J. M. (2020). Depth uncertainty in neural networks. In *Advances in Neural Information Processing Systems 33, NeurIPS*.

Antoran, J., Barbano, R., Leuschner, J., Hernández-Lobato, J. M., and Jin, B. (2023). Uncertainty estimation for computed tomography with a linearised deep image prior. *Transactions on Machine Learning Research, TMLR*.

Antorán, J., Bhatt, U., Adel, T., Weller, A., and Hernández-Lobato, J. M. (2021). Getting a CLUE: A method for explaining uncertainty estimates. In *9th International Conference on Learning Representations, ICLR*.

Antorán, J., Janz, D., Allingham, J. U., Daxberger, E. A., Barbano, R., Nalisnick, E. T., and Hernández-Lobato, J. M. (2022). Adapting the linearised laplace model evidence for modern deep learning. *Proceedings of the 39th International Conference on Machine Learning, ICML*.

Antorán, J. and Miguel, A. (2019). Disentangling and learning robust representations with natural clustering. In *18th IEEE International Conference On Machine Learning And Applications, ICMLA*.

Antorán, J., Padhy, S., Barbano, R., Nalisnick, E., Janz, D., and Hernández-Lobato, J. M. (2023). Sampling-based inference for large linear models, with application to linearised laplace. In *11th International Conference on Learning Representations, ICLR*.

Antorán, J., Allingham, J. U., and Hernández-Lobato, J. M. (2020). Variational depth search in resnets.

Antun, V., Renna, F., Poon, C., Adcock, B., and Hansen, A. C. (2020). On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proc. Nat. Acad. Sci.*

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*.

Arridge, S., Maaß, P., Öktem, O., and Schönlieb, C.-B. (2019). Solving inverse problems using data-driven models. *Acta Numer.*

Artemev, A., Burt, D. R., and van der Wilk, M. (2021). Tighter bounds on the log marginal likelihood of gaussian process regression using conjugate gradients. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. (2020). Pitfalls of in-domain uncertainty estimation and ensembling in deep learning.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI*.

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint: 1607.06450*.

Bach, F. R. (2014). Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *Journal of Machine Learning Research, JMLR*.

Baguer, D. O., Leuschner, J., and Schmidt, M. (2020). Computed tomography reconstruction using deep image prior and learned reconstruction methods. *Inverse Problems*.

Baragatti, M. and Pommeret, D. (2012). A study of variable selection using g-prior distribution with ridge parameter. *Computational Statistics & Data Analysis*.

Barbano, R., Antorán, J., Hernández-Lobato, J. M., and Jin, B. (2022a). A probabilistic deep image prior over image space. In *4th Symposium on Advances in Approximate Bayesian Inference, AABI*.

Barbano, R., Antorán, J., Leuschner, J., Hernández-Lobato, J. M., Jin, B., and Kereta, Z. (2023). Image reconstruction via deep image prior subspaces. *Transactions on Machine Learning Research, TMLR*.

Barbano, R., Leuschner, J., Antorán, J., Hernández-Lobato, J. M., and Jin, B. (2022b). Bayesian experimental design for computed tomography with the linearised deep image prior. *ICML Workshop on Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

Barbano, R., Leuschner, J., Schmidt, M., Denker, A., Hauptmann, A., Maass, P., and Jin, B. (2022c). An educated warm start for deep image prior-based micro ct reconstruction. *IEEE Transactions on Computational Imaging*.

Barutcu, S., Gürsoy, D., and Katsaggelos, A. K. (2022). Compressive ptychography using deep image and generative priors.

Becker, S. and LeCun, Y. (1989). Improving the convergence of back-propagation learning with second-order methods.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*.

Bernstein, S. (1946). *The Theory of Probabilities*. Gostechizdat.

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melancon, G. G., Krishnan, R., Stanley, J., Tickoo, O., Nachman, L., Chunara, R., Srikumar, M., Weller, A., and Xiang, A. (2021). Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*.

Bishop, C. and Tipping, M. (2003). *Bayesian Regression and Classification*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Blanchard, G. and Krämer, N. (2010). Optimal learning rates for kernel conjugate gradient regression. In *Advances in Neural Information Processing Systems 23, NeurIPS*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 31st International Conference on Machine Learning, ICML*.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*.

Bové, D. S. and Held, L. (2011). Hyper-g priors for generalized linear models. *Bayesian Analysis*.

Boyd, S. P. and Vandenberghe, L. (2014). *Convex Optimization*. Cambridge University Press.

Brock, A., De, S., and Smith, S. L. (2021a). Characterizing signal propagation to close the performance gap in unnormalized resnets. In *9th International Conference on Learning Representations, ICLR*.

Brock, A., De, S., Smith, S. L., and Simonyan, K. (2021b). High-performance large-scale image recognition without normalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Burger, M., Hauptmann, A., Helin, T., Hyvönen, N., and Puska, J.-P. (2021). Sequentially optimized projections in x-ray imaging. *Inverse Problems*.

Cai, Y., Li, Q., and Shen, Z. (2019). A quantitative analysis of the effect of batch normalization on gradient descent. In *Proceedings of the 35th International Conference on Machine Learning, ICML*.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *The 12th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: A review. *Statistical Science*.

Chambolle, A., Caselles, V., Cremers, D., Novaga, M., and Pock, T. (2010). An introduction to total variation for image analysis.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. (2020). Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in Neural Information Processing Systems 33, NeurIPS*.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. (2022). Gaussian process parameter estimation using mini-batch stochastic gradient descent: Convergence guarantees and empirical benefits. *Journal of Machine Learning Research, JMLR*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint: 2107.03374*.

Cheng, Z., Gadelha, M., Maji, S., and Sheldon, D. (2019). A Bayesian perspective on the deep image prior. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature.

Collaboration, T. E. H. T., Akiyama, K., Algaba, J. C., Alberdi, A., Alef, W., Anantua, R., Asada, K., Azulay, R., Baczko, A.-K., Ball, D., Baloković, M., Barrett, J., Benson, B. A., Bintley, D., Blackburn, L., Blundell, R., Boland, W., Bouman, K. L., Bower, G. C., Boyce,

H., Bremer, M., Brinkerink, C. D., Brissenden, R., Britzen, S., Broderick, A. E., Broguiere, D., Bronzwaer, T., Byun, D.-Y., Carlstrom, J. E., Chael, A., kwan Chan, C., Chatterjee, S., Chatterjee, K., Chen, M.-T., Chen, Y., Chesler, P. M., Cho, I., Christian, P., Conway, J. E., Cordes, J. M., Crawford, T. M., Crew, G. B., Cruz-Osorio, A., Cui, Y., Davelaar, J., Laurentis, M. D., Deane, R., Dempsey, J., Desvignes, G., Dexter, J., Doeleman, S. S., Eatough, R. P., Falcke, H., Farah, J., Fish, V. L., Fomalont, E., Ford, H. A., Fraga-Encinas, R., Freeman, W. T., Friberg, P., Fromm, C. M., Fuentes, A., Galison, P., Gammie, C. F., García, R., Gentaz, O., Georgiev, B., Goddi, C., Gold, R., Gómez, J. L., Gómez-Ruiz, A. I., Gu, M., Gurwell, M., Hada, K., Haggard, D., Hecht, M. H., Hesper, R., Ho, L. C., Ho, P., Honma, M., Huang, C.-W. L., Huang, L., Hughes, D. H., Ikeda, S., Inoue, M., Issaoun, S., James, D. J., Jannuzi, B. T., Janssen, M., Jeter, B., Jiang, W., Jimenez-Rosales, A., Johnson, M. D., Jorstad, S., Jung, T., Karami, M., Karuppusamy, R., Kawashima, T., Keating, G. K., Kettenis, M., Kim, D.-J., Kim, J.-Y., Kim, J., Kim, J., Kino, M., Koay, J. Y., Kofuji, Y., Koch, P. M., Koyama, S., Kramer, M., Kramer, C., Krichbaum, T. P., Kuo, C.-Y., Lauer, T. R., Lee, S.-S., Levis, A., Li, Y.-R., Li, Z., Lindqvist, M., Lico, R., Lindahl, G., Liu, J., Liu, K., Liuzzo, E., Lo, W.-P., Lobanov, A. P., Loinard, L., Lonsdale, C., Lu, R.-S., MacDonald, N. R., Mao, J., Marchili, N., Markoff, S., Marrone, D. P., Marscher, A. P., Martí-Vidal, I., Matsushita, S., Matthews, L. D., Medeiros, L., Menten, K. M., Mizuno, I., Mizuno, Y., Moran, J. M., Moriyama, K., Moscibrodzka, M., Müller, C., Musoke, G., Mejías, A. M., Michalik, D., Nadolski, A., Nagai, H., Nagar, N. M., Nakamura, M., Narayan, R., Narayanan, G., Natarajan, I., Nathanail, A., Neilsen, J., Neri, R., Ni, C., Noutsos, A., Nowak, M. A., Okino, H., Olivares, H., Ortiz-León, G. N., Oyama, T., Özel, F., Palumbo, D. C. M., Park, J., Patel, N., Pen, U.-L., Pesce, D. W., Piétu, V., Plambeck, R., PopStefanija, A., Porth, O., Pötzl, F. M., Prather, B., Preciado-López, J. A., Psaltis, D., Pu, H.-Y., Ramakrishnan, V., Rao, R., Rawlings, M. G., Raymond, A. W., Rezzolla, L., Ricarte, A., Ripperda, B., Roelofs, F., Rogers, A., Ros, E., Rose, M., Roshanineshat, A., Rottmann, H., Roy, A. L., Ruszczyk, C., Rygl, K. L. J., Sánchez, S., Sánchez-Arguelles, D., Sasada, M., Savolainen, T., Schloerb, F. P., Schuster, K.-F., Shao, L., Shen, Z., Small, D., Sohn, B. W., SooHoo, J., Sun, H., Tazaki, F., Tetarenko, A. J., Tiede, P., Tilanus, R. P. J., Titus, M., Toma, K., Torne, P., Trent, T., Traianou, E., Trippe, S., van Bemmel, I., van Langevelde, H. J., van Rossum, D. R., Wagner, J., Ward-Thompson, D., Wardle, J., Weintroub, J., Wex, N., Wharton, R., Wielgus, M., Wong, G. N., Wu, Q., Yoon, D., Young, A., Young, K., Younsi, Z., Yuan, F., Yuan, Y.-F., Zensus, J. A., Zhao, G.-Y., Zhao, S.-S., and Collaboration, T. E. H. T. (2021). First m87 event horizon telescope results. vii. polarization of the ring. *The Astrophysical Journal Letters*.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14, NeurIPS*.

Cox, R. T. (1946). Probability, Frequency and Reasonable Expectation. *American Journal of Physics*.

Cui, J., Gong, K., Guo, N., Wu, C., Kim, K., Liu, H., and Li, Q. (2021). Populational and individual information based PET image denoising using conditional unsupervised learning. *Phys. Med. & Biol.*

Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F., and Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. *Advances in Neural Information Processing Systems 27, NeurIPS*.

Darestani, M. Z. and Heckel, R. (2021). Accelerated MRI with un-trained neural networks. *IEEE Trans. Comput. Imag.*

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021a). Laplace redux–effortless Bayesian deep learning. In *Advances in Neural Information Processing Systems 34, NeurIPS*.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. (2020). Expressive yet tractable bayesian deep learning via subnetwork inference. In *2nd Symposium on Advances in Approximate Bayesian Inference, AABI*.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernandez-Lobato, J. M. (2021b). Bayesian deep learning via subnetwork inference. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Daxberger, E. A., Nalisnick, E. T., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. (2021c). Bayesian deep learning via subnetwork inference. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

de G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. In *6th International Conference on Learning Representations, ICLR*.

de G. Matthews, A. G., Hron, J., Turner, R. E., and Ghahramani, Z. (2017). Sample-then-optimize posterior sampling for bayesian linear models. In *1st Symposium on Advances in Approximate Bayesian Inference, AABI*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*.

Deng, Z., Zhou, F., and Zhu, J. (2022). Accelerated linearized laplace approximation for bayesian deep learning. In *Advances in Neural Information Processing Systems 35, NeurIPS*.

Der Sarkissian, H., Lucka, F., van Eijnatten, M., Colacicco, G., Coban, S. B., and Batenburg, K. J. (2019). Cone-Beam X-Ray CT Data Collection Designed for Machine Learning: Samples 1-8.

Dieuleveut, A., Flammarion, N., and Bach, F. R. (2017). Harder, better, faster, stronger convergence rates for least-squares regression. *Journal of Machine Learning Research, JMLR*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR*.

Draper, D. and Krnjajic, M. (2010). Calibration results for bayesian model specification. *Bayesian Analysis*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dusenberry, M. W., Jerfel, G., Wen, Y., Ma, Y.-a., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. (2020). Efficient and scalable bayesian neural nets with rank-1 factors. *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Elahi, M., Ricci, F., and Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*.

Elbakri, I. A. and Fessler, J. A. (2003). Efficient and accurate likelihood for iterative image reconstruction in x-ray computed tomography. In *Medical Imaging 2003: Image Processing*.

Eschenhagen, R., Daxberger, E., Hennig, P., and Kristiadi, A. (2021). Mixtures of laplace approximations for improved post-hoc uncertainty in deep learning. *arXiv preprint: 2111.03577*.

Feng, C. (2015). *Optimal Bayesian experimental design in the presence of model error*. PhD thesis, Massachusetts Institute of Technology.

Flamich, G. (2019). *Compression without Quantization*. PhD thesis, University of Cambridge.

Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019a). In-between uncertainty in bayesian neural networks. *ICML Workshop on Uncertainty and Robustness in Deep Learning*.

Foong, A. Y. K., Burt, D. R., Li, Y., and Turner, R. E. (2020). On the expressiveness of approximate inference in Bayesian neural networks. In *Advances in Neural Information Processing Systems 33, NeurIPS*.

Foong, A. Y. K., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019b). 'In-between' uncertainty in Bayesian neural networks. *arXiv preprint: 1906.11537*.

Foresee, F. D. and Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *International Conference on Neural Networks*.

Fortuin, V., Garriga-Alonso, A., Wenzel, F., Ratsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. (2021). Bayesian neural network priors revisited. In *3rd Symposium on Advances in Approximate Bayesian Inference, AABI*.

Foster, A. E. (2021). *Variational, Monte Carlo and Policy-Based Approaches to Bayesian Experimental Design*. PhD thesis, University of Oxford.

Fridman, L., Ding, L., Jenik, B., and Reimer, B. (2019). Arguing machines: Human supervision of black box ai systems that make life-critical decisions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Friston, K. J., Mattout, J., Trujillo-Barreto, N. J., Ashburner, J., and Penny, W. D. (2007). Variational free energy and the Laplace approximation. *NeuroImage*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*.

García-Ortegón, M., Simm, G. N. C., Tripp, A. J., Hernández-Lobato, J. M., Bender, A., and Bacallado, S. (2022). Dockstring: Easy molecular docking yields better benchmarks for ligand design. *Journal of Chemical Information and Modeling*.

Gardner, J. R., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems 31, NeurIPS*.

Garriga-Alonso, A., Aitchison, L., and Rasmussen, C. E. (2019). Deep convolutional networks as shallow Gaussian processes. In *7th International Conference on Learning Representations, ICLR*.

Geffner, T., Antorán, J., Foster, A., Gong, W., Ma, C., Kiciman, E., Sharma, A., Lamb, A., Kukla, M., Pawlowski, N., Allamanis, M., and Zhang, C. (2022). Deep end-to-end causal inference.

Germain, P., Bach, F. R., Lacoste, A., and Lacoste-Julien, S. (2016). Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems 29, NeurIPS*.

Ghosal, S. and van der Vaart, A. (2017). *Fundamentals of Nonparametric Bayesian Inference*. Cambridge University Press.

Gibbs, M. N. and MacKay, D. J. C. (1996). Efficient implementation of Gaussian processes for interpolation.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*.

Goddard, J. (2023). Hallucinations in ChatGPT: A cautionary tale for biomedical researchers. *Am J Med*.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*.

Gong, K., Catana, C., Qi, J., and Li, Q. (2019). PET image reconstruction using deep image prior. *IEEE Trans. Med. Imag.*

Graczykowski, L. K., Jakubowska, M., Deja, K. R., and Kabus, M. (2022). Using Machine Learning for Particle Identification in ALICE. *Jinst*.

Graves, A. (2011). Practical variational inference for neural networks.

Grünwald, P. (2004). A tutorial introduction to the minimum description length principle. *arXiv preprint: 0406077*.

Gull, S. F. (1988). *Bayesian Inductive Inference and Maximum Entropy*. Springer Netherlands.

Gull, S. F. (1989). *Bayesian Data Analysis: Straight-line fitting*.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *American Chemical Society Central Science*.

Halko, N., Martinsson, P., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*

Hansson, S. O. (2011). *Decision Theory: An Overview*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*.

Heckel, R. and Hand, P. (2019). Deep decoder: Concise image representations from untrained non-convolutional networks. In *7th International Conference on Learning Representations, ICLR*.

Heeger, D. (2000). Poisson model of spike generation.

Helin, T., Hyvönen, N., and Puska, J. (2022a). Edge-promoting adaptive Bayesian experimental design for x-ray imaging. *SIAM J. Sci. Comput.*

Helin, T., Hyvönen, N., and Puska, J.-P. (2022b). Edge-promoting adaptive Bayesian experimental design for X-ray imaging. *SIAM J. Sci. Comput.*

Hendrycks, D. and Gimpel, K. (2017). A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *5th International Conference on Learning Representations, ICLR*.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. (2020). Scaling laws for autoregressive generative modeling. *arXiv preprint: 2010.14701*.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, UAI*.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proceedings of the 31st International Conference on Machine Learning, ICML*.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Bui, T. D., Hernández-Lobato, D., and Turner, R. E. (2016). Black-box alpha divergence minimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems 27, NeurIPS*.

Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. (2017). Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*.

Hoffer, E., Banner, R., Golan, I., and Soudry, D. (2018). Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems 31, NeurIPS*.

Hoffman, Y. (2009). *Gaussian Fields and Constrained Simulations of the Large-Scale Structure*.

Hoffman, Y. and Ribak, E. (1991). Constrained realizations of Gaussian fields: a simple algorithm. *Astrophys. J. Lett.*

Hofmann, T., Schölkopf, B., and Smola, A. (2006). A tutorial review of rkhs methods in machine learning.

Hron, J., Matthews, A., and Ghahramani, Z. (2018). Variational Bayesian dropout: pitfalls and fixes. In *Proceedings of the 34th International Conference on Machine Learning, ICML*.

Hu, A., Corrado, G., Griffiths, N., Murez, Z., Gurau, C., Yeo, H., Kendall, A., Cipolla, R., and Shotton, J. (2022). Model-based imitation learning for urban driving. In *Advances in Neural Information Processing Systems 35, NeurIPS*.

Hutchinson, M. (1990). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Commun. Stat. Simul. Comput.*

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan, M. E. (2021a). Scalable marginal likelihood estimation for model selection in deep learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Immer, A., Korzepa, M., and Bauer, M. (2021b). Improving predictions of bayesian neural nets via local linearization. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Immer, A., Palumbo, E., Marx, A., and Vogt, J. E. (2023a). Effective bayesian heteroscedastic regression with deep neural networks. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

Immer, A., van der Ouderaa, T. F., Ratsch, G., Fortuin, V., and van der Wilk, M. (2022). Invariance learning in deep neural networks with differentiable laplace approximations. In *Advances in Neural Information Processing Systems 35, NeurIPS*.

Immer, A., Van Der Ouderaa, T. F. A., Van Der Wilk, M., Ratsch, G., and Schölkopf, B. (2023b). Stochastic marginal likelihood gradients using neural tangent kernels. In *Proceedings of the 39th International Conference on Machine Learning, ICML*.

Ioffe, S. (2010). Improved consistent sampling, weighted minhash and l1 sketching. In *International Conference on Data Dining*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 31st International Conference on Machine Learning, ICML*.

Ito, K. and Jin, B. (2014). *Inverse problems: Tikhonov theory and algorithms*. World Scientific.

Jacot, A., Hongler, C., and Gabriel, F. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31, NeurIPS*.

Janz, D., Hron, J., Mazur, P., Hofmann, K., Hernández-Lobato, J. M., and Tschiatschek, S. (2019). Successor uncertainties: Exploration and uncertainty in temporal difference learning. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Jaynes, E. and Justice, J. H. (1986). *Bayesian Methods: General Background*.

Jeffreys, H. (1939). *Theory of Probability*. Clarendon Press.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2024). Mixtral of experts.

Journel, A. G. and Huijbregts, C. J. (1978). *Mining geostatistics / [by] A. G. Journel and Ch. J. Huijbregts*. Academic Press London ; New York.

Kaipio, J. and Somersalo, E. (2005). *Statistical and computational inverse problems*. Springer-Verlag, New York.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*.

Khan, M. E., Immer, A., Abedi, E., and Korzepa, M. (2019a). Approximate inference turns deep networks into Gaussian processes. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *Proceedings of the 34th International Conference on Machine Learning, ICML*.

Khan, M. E. and Rue, H. (2023). The bayesian learning rule. *Journal of Machine Learning Research, JMLR*.

Khan, M. E. E. (2014). Decoupled variational gaussian inference. In *Advances in Neural Information Processing Systems 27, NeurIPS*.

Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. (2019b). Approximate inference turns deep networks into gaussian processes. *Advances in Neural Information Processing Systems 32, NeurIPS*.

Knopp, T. and Grosser, M. (2021). Warmstart approach for accelerating deep image prior reconstruction in dynamic tomography.

Kompa, B., Snoek, J., and Beam, A. L. (2021). Second opinion needed: communicating uncertainty in medical machine learning. *NPJ Digital Medicine*.

Kristiadi, A., Hein, M., and Hennig, P. (2020). Being bayesian, even just a bit, fixes overconfidence in relu networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Kunstner, F., Hennig, P., and Balles, L. (2019). Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30, NeurIPS*.

Laves, M.-H., Tölle, M., and Ortmaier, T. (2020). Uncertainty estimation in medical image denoising with bayesian deep image prior.

Lawrence, N. D. (2000). *Variational inference in probabilistic models*. PhD thesis, University of Cambridge.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1996). Effiicient backprop. In *Neural Networks: Tricks of the Trade*.

LeCun, Y., Simard, P., and Pearlmutter, B. (1992). Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *Advances in Neural Information Processing Systems 5, NeurIPS*.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Li, R., John, S. T., and Solin, A. (2023). Improving hyperparameter learning under approximate inference in gaussian process models. In *Proceedings of the 39th International Conference on Machine Learning, ICML*.

Li, Y. (2018). *Approximate Inference: New Visions*. PhD thesis, University of Cambridge.

Li, Z., Lyu, K., and Arora, S. (2020). Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. In *Advances in Neural Information Processing Systems 33, NeurIPS*.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008). Mixtures of g priors for bayesian variable selection. *Journal of the American Statistical Association*.

Lin, J. A., Antoran, J., and Hernández-Lobato, J. M. (2023a). Online laplace model selection revisited. In *5th Symposium on Advances in Approximate Bayesian Inference, AABI*.

Lin, J. A., Antoran, J., Padhy, S., Janz, D., Hernández-Lobato, J. M., and Terenin, A. (2023b). Sampling from gaussian process posteriors using stochastic gradient descent. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

Lin, J. A., Padhy, S., Antorán, J., Tripp, A., Terenin, A., Szepesvári, C., Hernández-Lobato, J. M., and Janz, D. (2024). Stochastic gradient descent for gaussian processes done right. In *12th International Conference on Learning Representations, ICLR*.

Liu, J., Sun, Y., Xu, X., and Kamilov, U. S. (2019). Image restoration using total variation regularized deep image prior. In *Icassp 2019*.

Lobacheva, E., Kodryan, M., Chirkova, N., Malinin, A., and Vetrov, D. P. (2021). On the periodic behavior of neural network training with batch normalization and weight decay. In *Advances in Neural Information Processing Systems 34, NeurIPS*.

Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*.

MacKay, D. J. (1992a). A practical bayesian framework for backpropagation networks. *Neural computation*.

MacKay, D. J. C. (1992b). Bayesian Interpolation. *Neural Computation*.

Mackay, D. J. C. (1992a). *Bayesian Methods for Adaptive Models*. PhD thesis.

Mackay, D. J. C. (1992b). Information-based objective functions for active data selection. *Neural Computation*.

Mackay, D. J. C. (1996). Bayesian non-linear modeling for prediction competition. In *Maximum Entropy and Bayesian Methods*.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint: 1611.00712*.

Maddox, W., Tang, S., Moreno, P. G., Wilson, A. G., and Damianou, A. C. (2021). Fast adaptation with linearized neural networks. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Maddox, W. J., Benton, G. W., and Wilson, A. G. (2020). Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv preprint: 2003.02139*.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research, JMLR*.

Martens, J. (2014). New insights and perspectives on the natural gradient method. *arXiv preprint: 1412.1193*.

Martens, J. and Grosse, R. B. (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 31st International Conference on Machine Learning, ICML*.

Martinsson, P.-G. and Tropp, J. A. (2020). Randomized numerical linear algebra: Foundations and algorithms. *Acta Numer.*

Masegosa, A. R. (2020). Learning under model misspecification: Applications to variational and ensemble methods. In *Advances in Neural Information Processing Systems 33, NeurIPS*.

Matthews, A. G. d. G. (2017). *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge.

Matthews, A. G. d. G., Hensman, J., Turner, R., and Ghahramani, Z. (2016). On sparse variational methods and the kullback-leibler divergence between stochastic processes. *Journal of Machine Learning Research, JMLR*.

Midgley, L. I., Stimper, V., Antorán, J., Mathieu, E., Schölkopf, B., and Hernández-Lobato, J. M. (2023). SE(3) equivariant augmented coupling flows. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

Minka, T. (2000). Bayesian linear regression.

Minka, T. (2004). Power ep. Technical report.

Minka, T. (2007). The ep energy function and minimization schemes.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*.

Murray, C., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. (2022a). Addressing bias in active learning with depth uncertainty networks... or not. In *Proceedings on "I (Still) Can't Believe It's Not Better!" at NeurIPS 2021 Workshops*.

Murray, C., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. (2022b). Depth uncertainty networks for active learning.

Nalisnick, E., Gordon, J., and Miguel Hernandez-Lobato, J. (2021). Predictive complexity priors. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer.

Neal, R. M. and Hinton, G. E. (1998). *A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants*.

Neath, A. A. and Cavanaugh, J. E. (2012). The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o$(1/k^2)$. In *Doklady Akademii Nauk SSSR*.

Nielsen, S. F. (2000). The stochastic EM algorithm: estimation and asymptotic results. *Bernoulli*.

Novak, R., Sohl-Dickstein, J., and Schoenholz, S. S. (2022). Fast finite width neural tangent kernel. In *Proceedings of the 38th International Conference on Machine Learning, ICML*.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2019). Bayesian deep convolutional networks with many channels are Gaussian processes. In *7th International Conference on Learning Representations, ICLR*.

Ober, S. W. and Aitchison, L. (2021). Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Ober, S. W. and Rasmussen, C. E. (2019). Benchmarking the neural linear model for regression. *arXiv preprint: 1912.08416*.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. (2021). The promises and pitfalls of deep kernel learning. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence, UAI*.

Ongie, G., Jalal, A., Baraniuk, R. G., Metzler, C. A., Dimakis, A. G., and Willett, R. (2020). Deep learning techniques for inverse problems in imaging. *IEEE J. Sel. Areas Inform. Theory*.

Opper, M. and Winther, O. (2005). Expectation consistent approximate inference. *Journal of Machine Learning Research, JMLR*.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. (2019). Practical deep learning with bayesian principles. *arXiv preprint: 1906.02506*.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31, NeurIPS*.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Roy, B. V. (2023). Epistemic neural networks. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Lu, X., and Roy, B. V. (2022). Evaluating high-order predictive distributions in deep learning. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, UAI*.

Papandreou, G. and Yuille, A. L. (2010). Gaussian sampling by local perturbations. In *Advances in Neural Information Processing Systems 23, NeurIPS*.

Paulsen, V. I. and Raghupathi, M. (2016). *An introduction to the theory of reproducing kernel Hilbert spaces*. Cambridge University Press.

Pearce, T., Leibfried, F., and Brintrup, A. (2020). Uncertainty in neural networks: Approximately bayesian ensembling. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*.

Pearl, J. (1982). Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. (2021). Random feature attention. In *9th International Conference on Learning Representations, ICLR*.

Pinder, T. and Dodd, D. (2022). Gpjax: A gaussian process framework in jax. *Journal of Open Source Software*.

Pinzi, L. and Rastelli, G. (2019). Molecular docking: Shifting paradigms in drug discovery. *International Journal of Molecular Sciences*.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*.

Polyak, B. T. (1990). New stochastic approximation type procedures. *Avtomatika i Telemekhanika*.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: the Art of Scientific Computing*. Cambridge University Press.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20, NeurIPS*.

Rasmussen, C. and Ghahramani, Z. (2000). Occam's razor. In *Advances in Neural Information Processing Systems 13, NeurIPS*.

Reid, I., Choromanski, K., Berger, E., and Weller, A. (2024). Universal graph random features. In *12th International Conference on Learning Representations, ICLR*.

Reid, I., Choromanski, K. M., Likhosherstov, V., and Weller, A. (2023). Simplex random features. In *Proceedings of the 39th International Conference on Machine Learning, ICML*.

Rényi, A. (1961). On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*.

Ritter, H., Botev, A., and Barber, D. (2018). A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR*.

Rogers, D. and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*.

Rubens, N., Elahi, M., Sugiyama, M., and Kaplan, D. (2015). Active learning in recommender systems. *Recommender Systems Handbook*.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747*.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D*.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Ruppert, D. (1988). Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems 35, NeurIPS*.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *The 12th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Saul, L. and Jordan, M. (1998). *A Mean Field Learning Algorithm for Unsupervised Neural Networks*.

Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Schraudolph, N. N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Comput*.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *Computational Learning Theory*.

Seeger, M. W. (2009). On the submodularity of linear experimental design.

Seeger, M. W. and Nickisch, H. (2011). Large scale bayesian inference and experimental design for sparse linear models. *SIAM J. Imaging Sci.*

Shalev-Shwartz, S. and Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research, JMLR*.

Shen, Y., Daheim, N., Cong, B., Nickl, P., Marconi, G. M., Bazan, C., Yokota, R., Gurevych, I., Cremers, D., Khan, M. E., and Möllenhoff, T. (2024). Variational learning is effective for large deep networks.

Shen, Z., Wang, Y., Wu, D., Yang, X., and Dong, B. (2022). Learning to scan: A deep reinforcement learning approach for personalized scanning in ct imaging. *Inverse Problems and Imaging*.

Sinharay, S. and Stern, H. S. (2002). On the sensitivity of bayes factors to the prior distributions. *The American Statistician*.

Skilling, J. (1989). *Classic Maximum Entropy*.

Smola, A. J. and Schölkopf, B. (1998). *Learning with Kernels*. MIT Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25, NeurIPS*.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. (2019a). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J. V., Ren, J., and Nado, Z. (2019b). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Stigler, S. M. (1986). Laplace's 1774 memoir on inverse probability. *Statistical Science*.

Stuart, A. M. (2010). Inverse problems: a Bayesian perspective. *Acta Numer.*

Sutherland, D. J. and Schneider, J. G. (2015). On the error of random fourier features. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence, UAI*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML*.

Terenin, A. (2022). *Gaussian Processes and Statistical Decision-making in Non-Euclidean Spaces*. PhD thesis, Imperial College London.

Terenin, A., Burt, D. R., Artemev, A., Flaxman, S., van der Wilk, M., Rasmussen, C. E., and Ge, H. (2023). Numerically stable sparse gaussian processes via minimum separation using cover trees. *Journal of Machine Learning Research, JMLR*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*.

Tian, Y., Zhang, Y., and Zhang, H. (2023). Recent advances in stochastic gradient descent in deep learning. *Mathematics*.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of ill-posed problems*. V. H. Winston & Sons.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research, JMLR*.

Titsias, M. K. (2009a). Variational learning of inducing variables in sparse gaussian processes. In *The 12th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Titsias, M. K. (2009b). Variational model selection for sparse gaussian process regression. Technical report, University of Manchester.

Titsias, M. K. and Ruiz, F. J. R. (2019). Unbiased implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*.

Tölle, M., Laves, M., and Schlaefer, A. (2021). A mean-field variational inference approach to deep image prior for inverse problems in medical imaging. In *Medical Imaging with Deep Learning, 7-9 July 2021, Lübeck, Germany*.

Tripp, A., Bacallado, S., Singh, S., and Hernández-Lobato, J. M. (2023). Tanimoto random features for scalable molecular machine learning. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

Trott, O. and Olson, A. J. (2010). Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*.

Turner, R. E. and Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018a). Deep image prior. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2018b). Deep image prior. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2020). Deep image prior. *Int. J. Comput. Vis.*

Uppal, A., Stensbo-Smidt, K., Boomsma, W., and Frellsen, J. (2023). Implicit variational inference for high-dimensional posteriors. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

van der Ouderaa, T. F. A., Immer, A., and van der Wilk, M. (2023). Learning layer-wise equivariances automatically using gradients. In *Advances in Neural Information Processing Systems 36, NeurIPS*.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional gaussian processes. In *Advances in Neural Information Processing Systems 30, NeurIPS*.

van Laarhoven, T. (2017). L2 regularization versus batch and weight normalization. *arXiv preprint: 1706.05350*.

Vapnik, V. (1995). *The Nature of Statistical Learning*. Springer.

Varre, A. V., Pillaud-Vivien, L., and Flammarion, N. (2021). Last iterate convergence of sgd for least-squares in the interpolation regime. *Advances in Neural Information Processing Systems 34, NeurIPS*.

Vasconcelos, F., He, B., Singh, N., and Teh, Y. W. (2022). UncertaINR: Uncertainty quantification of end-to-end implicit neural representations for computed tomography.

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*.

Wang, G., Ye, J. C., and De Man, B. (2020). Deep learning for tomographic image reconstruction. *Nature Mach. Intell.*

Wang, H., Li, T., Zhuang, Z., Chen, T., Liang, H., and Sun, J. (2021). Early stopping for deep image prior.

Wang, K. A., Pleiss, G., Gardner, J. R., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems 32, NeurIPS*.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. (2024). Improving text embeddings with large language models.

Weiser, B. and Schweber, N. (2023). The chatgpt lawyer explains himself. *The New York Times*.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 27th International Conference on Machine Learning, ICML*.

Wenzel, F., Roth, K., Veeling, B. S., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. (2020). How good is the bayes posterior in deep neural networks really? In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

West, M. (2018). Outlier Models and Prior Distributions in Bayesian Linear Regression. *Journal of the Royal Statistical Society: Series B (Methodological)*.

Wild, V., Kanagawa, M., and Sejdinovic, D. (2021). Connections and equivalences between the nystrom method and sparse variational gaussian processes. *arXiv preprint: 2106.01121*.

Wilkinson", W. J. ("2019"). *"Gaussian process modelling for audio signals"*. PhD thesis, "Queen Mary University of London".

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT press Cambridge, MA.

Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 31st International Conference on Machine Learning, ICML*.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Efficiently sampling functions from gaussian process posteriors. In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2021). Pathwise conditioning of gaussian processes. *Journal of Machine Learning Research, JMLR*.

Wipf, D. P. and Nagarajan, S. S. (2007). A new view of automatic relevance determination. In *Advances in Neural Information Processing Systems 20, NeurIPS*.

Wu, Y. and He, K. (2020). Group normalization. *Int. J. Comput. Vis.*

Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., et al. (2019). Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*.

Yang, Y., Yao, K., Repasky, M. P., Leswing, K., Abel, R., Shoichet, B. K., and Jerome, S. V. (2021). Efficient exploration of chemical space with docking and deep learning. *Journal of Chemical Theory and Computation*.

Yu, F. X., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., and Kumar, S. (2016). Orthogonal random features. In *Advances in Neural Information Processing Systems 29, NeurIPS*.

Zellner, A. (1986). *On assessing prior distributions and Bayesian regression analysis with g prior distributions*.

Zhang, H., Dauphin, Y. N., and Ma, T. (2019). Fixup initialization: Residual learning without normalization. In *7th International Conference on Learning Representations, ICLR*.

Zhu, Y., Yuan, H., Wang, S., Liu, J., Liu, W., Deng, C., Chen, H., Dou, Z., and Wen, J.-R. (2024). Large language models for information retrieval: A survey.

Zou, D., Wu, J., Braverman, V., Gu, Q., and Kakade, S. M. (2021). Benign overfitting of constant-stepsize SGD for linear regression. In *Conference on Learning Theory*.

Appendix A

Experimental setup details for Chapter 5

Here, we provide the details of our experimental setup which were omitted from the main text.

A.1 Experiments with full Hessian computation

This subsection concerns the experiments which use small architectures for which exact Hessian computation is tractable. These experiments are described in Section 5.5.1 and Section 5.5.2 of the main text. We first describe the setup components shared among architectures and then provide architecture-specific details. We exclude details for the U-net used in Section 5.5.2. Instead we provide these together with a brief description of the tomographic reconstruction task it performs in Section A.2.

Unless specified otherwise, NN weights $\tilde{v}$ are learnt using SGD, with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 1×10^{-4}. We trained for 90 epochs, using a multi-step LR scheduler with a decay rate of 0.1 applied at epochs 40 and 70. This is a standard choice for CNNs and is default in the examples provided by Pytorch.

The linear weights $w_\star$ are optimised using Adam and with their gradients calculated using algorithm 2. We use a learning rate of 1×10^{-4} and train for 100 epochs. We set the initial regularisation parameter to be isotropic $A = aI$ with $a = 1 \times 10^{-4}$.

A.1.1 CNN

Our CNN is based on the LeNet architecture with a few variations found in more modern neural networks. The architecture contains 3 convolutional blocks, followed by global average pooling in the spatial dimensions, a flatten operation, and finally a fully-connected layer. The convolutional blocks consist of Conv $\rightarrow$ ReLU $\rightarrow$ BatchNorm. Instead of using max pooling layers, as in the original LeNet variants, we use convolutions with a stride of 2. The first convolution is 5×5, while the next two are 3×3. As described in the main text, we consider architectures of 3 different sizes. Table A.1 shows the number of the filters and number of parameters for each size of this model. The *Big* model's values where chosen to create a model as large as possible while keeping full-covariance Laplace inference tractable on one A100 GPU.

	Conv1 Filters	Conv2 Filters	Conv3 Filters	Params.	Hessian Size
Big	42	48	60	46 024	15.68 GB
Med	32	32	64	29 226	6.36 GB
Small	16	32	32	14 634	1.60 GB

Table A.1 Architecture parameters for the CNNs used in experiments.

A.1.2 ResNet, Pre-ResNet, and Biased-ResNet

Our ResNet is based on our CNN architecture. We replace the second and third convolutional blocks with residual blocks. The main branch of the residual blocks consist of Conv $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ Conv $\rightarrow$ BatchNorm. We apply a final ReLU layer after the residual is added. All of the convolutions in the residual blocks use the same number of filters. In order to downsample our features between blocks we use 1×1 convolutions with a stride of 2. Table A.2 shows the number of the filters and number of parameters used for each size of this model.

Our Pre-ResNet architecture is identical to the ResNet except the main branch cosists of BatchNorm $\rightarrow$ ReLU $\rightarrow$ Conv $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ Conv, and we do not apply a ReLU after adding the residual.

Note that the standard ResNet architecture does not apply biases in the convolution layers. The only biases in the entire network are placed in the dense output layer. For our experiment where biases are included in the Jacobian feature expansion in Section 5.5.1, we modify the ResNet architecture to include biases in all convolutional layers in addition to the already

	Conv1 Filters	ResBlock 1 Filters	ResBlock 2 Filters	Params.	Hessian Size
Big	22	42	64	45 576	15.48 GB
Med	16	32	64	26 874	5.38 GB
Small	12	24	32	14 814	1.64 GB

Table A.2 Architecture parameters for the ResNets used in experiments.

present final dense layer bias. These biases account for a small increase in parameters, to 14 898, 26 986, and 45 726, in the small, medium, and big cases, respectively.

A.1.3 FixUp ResNet

Our FixUp-ResNet architecture follows the standard ResNet structure described above, with the additional FixUP offsets and multipliers described in (Zhang et al., 2019). We also follow Zhang et al. (2019) in zero initialising the dense layer, and scaling the convolution weight initialisation as a function of the depth of the network.

When training FixUp-ResNets, we use the Adam optimiser with a fixed learning rate of 0.01.

A.1.4 Transformer

Our Transformer architecture contains two encoder layers with two attention heads each, and no dropout. Its input is a sequence of tokens, to which we apply a linear embedding. We add a learnable `class` embedding for each input. This `class` token is used to classify the input. We do not use positional encoding, preserving permutation invariance in the input. The sizes of the embeddings and the MLP hidden dimensions are provided in Table A.3.

When training Transformers, we use the Adam optimiser with a learning rate of 3×10^{-3}. We use an exponential learning rate decay with a gamma of 0.99 applied after every epoch of training.

	MLP Dim	Embedding Dim	Params.	Hessian Size
Big	120	50	45 900	15.70 GB
Med	80	40	27 090	5.47 GB
Small	60	30	15 520	1.79 GB

Table A.3 Architecture parameters for the Transformers used in experiments.

A.2 U-Net tomographic reconstruction of KMNIST digits

In this section, we provide experimental details for the tomographic reconstruction results in Section 5.5.2.

Our setup almost exactly replicates that of Barbano et al. (2022a) and Antoran et al. (2023), which form the basis of Chapter 7. We refer to this chapter for an introduction to tomographic reconstruction with the deep image prior.

We use 10 test images from the KMNIST dataset, which consists of 28×28 grey-scale images of Hiragana characters (Clanuwat et al., 2018), we simulate y with 20 angles taken uniformly from the range $0°$ to $180°$, and add 5% white noise to the projected inputs $\mathcal{T}x$. We reconstruct x using the Deep Image Prior (DIP) (Ulyanov et al., 2018b), which parametrises the reconstruction x as the output of a U-net $g(v)$ (Ronneberger et al., 2015).

We use the U-net like architecture deployed by Barbano et al. (2022c). Group norm is placed before every 3×3 convolution operation. The U-net architecture is a encoder-decoder, fully convolutional deep model constructing multi-level feature maps. We identify 3 distinct blocks for both the encoder branch and and 2 blocks for the the decoder branch: `In`, $\texttt{Down}_0$, $\texttt{Down}_1$, and $\texttt{Up}_0$ and $\texttt{Up}_1$, respectively. The `In` block consists of a 3×3 convolution. `Down` blocks consist of a 3×3 convolution with stride of 2 followed by a 3×3 convolution operation and a bi-linear up-sampling. The `Up` blocks instead consist of two successive 3×3 convolutional operations. Given the use of the leaky ReLU non-linearity, the normalised parameter groups of this network coincide with the described blocks. The number of channels is set to 32 at every scale. Multi-channel feature maps from the `In` block and from $\texttt{Down}_0$ are first transformed via a 1×1 convolutional operation to 4 channel feature maps and then fed to $\texttt{Up}_1$, $\texttt{Up}_0$. The reconstructed image is obtained as the output of $\texttt{Up}_1$ further processed via a 1×1 convolutional layer. The total number of parameters is 78k. This is too many for full Hessian construction on GPU but we get around this issue by performing inference in the lower dimensional space of observations, as described in Section 7.2.2 and Section 7.3.1. We refer to Antoran et al. (2023) for a full list of hyperparameters involved in training the U-net.

The prior covariance A^{-1} is a filter-wise block-diagonal matrix which applies separate regularisation to the parameters of each block in the U-net. This matches the prior described in Section 7.2.3, but without the Matérn covariance structure. For the single regulariser experiment, we keep the same prior structure but tie the marginal prior variance of all parameters. That is, we ensure all entries of the diagonal of A^{-1} are the same. The parameters of these regularisers are learnt via model evidence optimisation, as described in the main text.

A.3 Large scale experiments

For scaling linearised Laplace to ResNet-50 with 25M parameters, we employ a Kronecker-factorisation of the Hessian/GGN. This is a common way to scale the Laplace approximation to large models (Daxberger et al., 2021a) and was originally proposed in Ritter et al. (2018). We use the recently-released `laplace` library[1] (Daxberger et al., 2021a) for fitting the KFAC Laplace models. For ResNets with batch norm, we use the reference implementation from the `torchvision` package[2]. For ResNets with fixUp, we use a popular open-source implementation[3]. To train the ResNet parameters, which will be used as the linearisation points $\tilde{v}$, we use the same hyperparameters as described at the top of Section A.1 for both batch norm and FixUp ResNets.

[1] https://github.com/AlexImmer/Laplace
[2] https://pytorch.org/vision/stable/models.html
[3] https://github.com/hongyi-zhang/FixUp