Broadening the Scope of Gaussian Processes for Large-Scale Learning

Kurt Cutajar

This dissertation is submitted for the degree of
Doctor of Philosophy
in the Doctoral School N°130:
Computer Science, Telecommunications and Electronics of Paris
of the Sorbonne University

Committee in Charge:

MAURIZIO FILIPPONE	EURECOM	ADVISOR
ROBERT JENSSEN	ARCTIC UNIVERSITY OF NORWAY	REVIEWER
DINO SEJDINOVIC	UNIVERSITY OF OXFORD	REVIEWER
PIETRO MICHIARDI	EURECOM	EXAMINER
SERENA VILLATA	LABORATOIRE I3S/CNRS	EXAMINER

April 2019

I dedicate this thesis to my brother Dale.

Acknowledgements

I am pleased to finally have the opportunity to properly thank all those who contributed towards making this PhD such a remarkable experience. Above all, I am indebted to my supervisor Maurizio Filippone for persuading me to join him at EURECOM three years ago and embark on this journey. Maurizio enthusiastically shared his ideas, patiently explained concepts I was unfamiliar with, and when the time was right, let me find my own feet and set my own pace. Through hard work and sheer determination, Maurizio transformed the humble machine learning duo we started off in late 2015 into a fully-fledged successful team, creating a stimulating environment for collaborating and sharing ideas. I sincerely appreciate his sustained mentoring and support at every step of the way, and hope our collaborations will not end here.

I am very grateful to Pietro Michiardi for not only immersing himself in extending the DGP work to operate on distributed frameworks, but for also being a grounded and supportive figure during these three years. His steely insistence was instrumental for allowing me to undertake both of my visiting periods away from EURECOM.

Many thanks to Mike Osborne for collaborating on the preconditioning work at the very beginning of my studies. Mike, along with Stephen Roberts, was kind enough to host me in his machine learning group in Oxford for three months in Summer 2017, which was both a formative and enjoyable experience. It was also through Mike that I got to meet and work with Jack Fitzsimons - besides becoming friends, I greatly enjoyed our breakneck, always-on collaboration, and the two papers we published together are a testament to the tunnel-vision commitment we had during those months. Collaborating on a paper with Diego Granziol also improved my appreciation and understanding of maximum entropy and other topics related to information theory.

Over the duration of the PhD, I had the pleasure of sharing three poster sessions with Edwin V. Bonilla for presenting the outcome of our collaborations, much of which has defined the overall scope of this thesis. Edwin's dedication, hands-on contribution, and

relentless high standards always inspired me to work harder. Likewise, I sincerely appreciate John P. Cunningham's involvement in my first paper, whose subsequent tips for writing and organising a convincing rebuttal influenced subsequent endeavours. I must also thank Philipp Hennig for inviting me to MPI in Tübingen for a week while I was working on probabilistic numerics, as well as Lukas Balles for organising the visit and being a good friend and host. Although we have yet to meet in person, I am also grateful to Karl Krauth for steering the work and development on AutoGP during our collaboration.

My internship at Amazon research in Cambridge was one of the highlights of the PhD, and I am especially grateful to Mark Pullin and Javier González for their helpful mentoring and for providing me the means to make the most of that experience. I also appreciate Andreas Damianou and Neil Lawrence's involvement in my work on multi-fidelity modelling, as well all other interns and members of the team who contributed towards making my four-month stay in Cambridge a memorable experience.

I will forever cherish the friendships made during these three years, starting with the roster of fellow PhD students and research scientists at EURECOM. I am also thankful for three generations of Masters students at EURECOM (too many to name individually) who always welcomed me in their midst, and provided the necessary moments of levity for enjoying my stay on the sunny French coast. Similarly, my stay in Oxford was made particularly pleasant by everyone within the machine learning sphere. In spite of my continued absence and busy schedule, both my family and friends back home in Malta, as well as friends made during my previous studies in Glasgow, always bore with my tight agenda and made the time to meet up and regularly stay in touch.

Finally, I cannot thank my parents enough for their patience, love, and support all throughout these roller-coaster three years. As with all my previous achievements, none of this would have been possible without their enduring encouragement and selfless understanding. Although it can never make up for all lost time, I dedicate this thesis to my dearest brother Dale.

Abstract

The renewed importance of decision making under uncertainty in practical applications of machine learning calls for a re-evaluation of Bayesian inference techniques targeting this goal in the big data regime. Gaussian processes (GPs) are a fundamental building block of many probabilistic kernel methods that can be applied in a large variety of modelling scenarios; however, the computational and storage complexity of GPs presents the primary barrier to their scaling to large modern datasets.

The contributions presented in this thesis are two-fold. We first tackle the problem of preserving good predictive performance while enabling GPs to be applied to larger datasets by proposing a novel scheme for accelerating regression and classification by way of preconditioned conjugate gradient. This approach is exact in the limit of iterations, and is consequently the best alternative to exact GP inference when Cholesky decompositions are no longer feasible. The presentation of this methodology is complemented by a thorough analysis of how to design and select suitable preconditioners for this purpose. In the spirit of probabilistic numerics, we also show how the numerical uncertainty introduced by formulations of GPs relying on approximated linear algebra should be adequately evaluated and sensibly incorporated.

The resilient appeal of models relying on Bayesian inference in the advent of competitive deep learning techniques can be attributed to their well-founded quantification of uncertainty. Bridging the gap between GPs and deep learning techniques remains a pertinent research goal, and the second broad contribution of this thesis is to establish and reinforce the role of GPs, and their deep counterparts (DGPs), in this setting. The AutoGP model outlined in this thesis sets a new standard for evaluating the performance of GPs in comparison to deep models, and the results obtained on several benchmark datasets are considered to be state-of-the-art among competing GP models. Meanwhile, the scarce use of DGPs within the wider machine learning body can be partly attributed to the fact that they are still widely regarded as interesting theoretical constructions with limited practical

appeal. In this thesis, we exploit the connection between neural networks and GPs by way of random feature expansions to develop a practical and scalable algorithm for learning DGPs. The proposed DGP model can handle significantly more layers than are usually assumed within the literature on DGPs, while also being scalable to large datasets for which DGP inference had previously been considered infeasible.

Preface

This thesis gathers, unifies, and extends the work carried out over the duration of the PhD, with particular emphasis on publications led by myself, but also highlighting co-authored work carried out in collaboration with others. All work featured here has been published in established, peer-reviewed conferences, an outline of which is given in this preamble.

Chapter 3 is based entirely upon the work presented in:

> **Kurt Cutajar**, Michael A. Osborne, John P. Cunningham, and Maurizio Filippone. *Preconditioning Kernel Matrices.* In Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York, USA.

while **Chapter 4** expands on:

> Jack Fitzsimons, **Kurt Cutajar**, Michael A. Osborne, Stephen Roberts, and Maurizio Filippone. *Bayesian Inference of Log Determinants.* In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia.

The original foundations for this work were conceived by Jack Fitzsimons, but its fruition was the outcome of close collaboration on aspects related to formulating the methodology, designing the experiments, and writing. The sections describing that work follow the original structure and presentation provided in the paper, but several concepts have been updated and rewritten in order to clarify the overall exposition. The remainder of the chapter, including all content related to combining probabilistic numerics with Gaussian processes, is novel to this manuscript. Chapter 4 also contains a brief mention of:

Jack Fitzsimons, Diego Granziol, **Kurt Cutajar**, Michael A. Osborne, Maurizio Filippone, and Stephen Roberts. *Entropic Trace Estimates for Log Determinants*. In Machine Learning and Knowledge Discovery in Databases - European Conference, ECML/PKDD 2017, Skopje, Macedonia.

but its contributions are not featured here.

Chapter 5 begins with a recap of the work presented in:

Karl Krauth, Edwin V. Bonilla, **Kurt Cutajar** and Maurizio Filippone. *AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models*. In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia.

This work started with an investigation I carried out on developing improved leave-one-out cross-validation objectives for optimising Gaussian processes. Karl Krauth and Edwin V. Bonilla took charge of extending these ideas in the ensuing collaboration. The brief exposition of this work is mostly adapted from the paper itself, with some elements being clarified and rewritten in order to suit the style of this thesis. The bulk of this chapter is however based on the work presented in:

Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. *Random Feature Expansions for Deep Gaussian Processes*. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, Australia.

with some elements taken from the associated workshop submission:

Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. *Accelerating Deep Gaussian Process Inference with Arc-cosine Kernels*. In First Bayesian Deep Learning workshop in Advances of Neural Information Processing Systems, NeurIPS 2016, Barcelona, Spain.

Finally, the following work carried out in relation to multi-fidelity modelling was not included in this thesis in order to preserve the overriding theme of developing scalable methodologies for Gaussian processes and their deep counterparts:

Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. *Deep Gaussian Processes for Multi-fidelity Modeling*. In Third Bayesian Deep Learning workshop in Advances of Neural Information Processing Systems, NeurIPS 2018, Montreal, Canada.

An extended version of this work is available on arXiv (1903.07320).

Table of Contents

List of Figures

Chapter 1

Uncertainty in Decision Making

The prevalence of machine learning in automating processes and carrying out tasks previously requiring human expertise has recently been faced with increased scrutiny and apprehension which is expected to steer the direction in which this field will develop in the coming years. Whereas groundbreaking innovations such as self-driving cars and automated medical diagnosis were once greeted as the way of the future (Bimbraw, 2015; Bostrom, 2014; Frey and Osborne, 2017), heavily-publicised incidents arising from the application of machine learning models to such domains have cast a heavy shadow over their robustness and reliability (Awad et al., 2018; Bonnefon et al., 2016). Although the cause of such deficiencies can be attributed to several factors, a persistent concern is that machine learning lacks the generalisation capabilities innate to human reasoning that enable greater adaptability and flexibility when affronted with a new task or unexpected setting. This highlights the importance of applying greater caution when relying on the raw outcomes of such models for critical decision making.

This shift in perspective has sparked a resurgence of interest in probabilistic modelling, which is inherently intended to capture the uncertainty or lack of knowledge a model might have about a learned task (Krzywinski and Altman, 2013). Let us consider a practical example for illustration - assume we are interested in developing a simple binary classifier for detecting viral infections in blood samples collected from patients, where the data available for training the model consists of either healthy samples or instances where the blood cells are already fully infected. In this setting, both a deterministic model and a probabilistic alternative are likely to make an erroneous prediction when presented with a blood sample where the virus is benign or still in its early stages. However, although

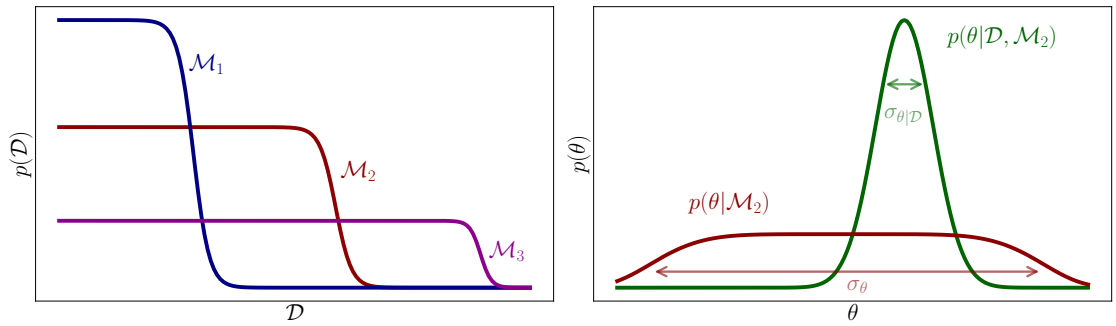
both models potentially return an incorrect result, the output obtained from the probabilistic model can be supplemented with an additional measure of the model's confidence or uncertainty in the prediction. In this example, high uncertainty estimates accompanying predictions can then be interpreted as an indicator for carrying out more rigorous tests on the patient or collecting more blood samples for retraining the model. Conversely, a deterministic classifier will produce a single binary outcome irrespective of how likely the prediction is to be correct, which may be unsatisfactory for high-risk or safety-critical applications such as health monitoring.

1.1 Bayesian Modelling

This division between modelling philosophies can often be traced back to the purported dichotomy between frequentist and Bayesian inference approaches. The frequentist view constructs a hypothesis relying exclusively on the data available at training time, regardless of any prior assumptions on the expected outcomes of the hypothesis. In practice, using techniques such as bootstrapping (Efron, 1979), predictions having the form of probability distributions can still be obtained by way of random and repeated treatments of the available data. However, this is often an ad hoc procedure. On the other hand, a more principled way of developing probabilistic models relies on Bayesian inference (de Finetti, 1974; Ghahramani, 2013), which is the foundation upon which the methods explored in this thesis are built. One of the primary motivations for using Bayesian inference is best described by way of its connection to the principle of Occam's razor (MacKay, 1991; Rasmussen and Ghahramani, 2000). In crude terms, Occam's razor denotes a preference for simpler explanations that sufficiently describe observations over more complex alternatives; in machine learning parlance, this can be interpreted as favouring models which generalise well as opposed to others that overfit the data available at training time. Bayes' theorem is formally given as

$$p\left(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_i\right) = \frac{p\left(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i\right) p\left(\boldsymbol{\theta}|\mathcal{M}_i\right)}{p\left(\mathcal{D}|\mathcal{M}_i\right)}, \tag{1.1}$$

where $\boldsymbol{\theta}$ denotes the parameters characterising a model $\mathcal{M}_i$ in a finite set of candidates $\mathcal{M}$, and $\mathcal{D}$ is the available data. The distribution $p\left(\boldsymbol{\theta}|\mathcal{M}_i\right)$ represents our *prior* beliefs on the values which the model's parameters are expected to take, while $p\left(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i\right)$ measures the *likelihood* of observing the available data using the designated model configuration. Dividing by the model *evidence* $p\left(\mathcal{D}|\mathcal{M}_i\right)$, which also serves as a normalising constant,



(a) Dataset fit for models with increasing complexity.

(b) Prior over $\boldsymbol{\theta}$ and posterior distribution after conditioning on data $\mathcal{D}$.

Fig. 1.1 Visualisation of the relationship between Occam's razor and Bayesian modelling. These plots are adapted from the exposition given in MacKay (2003) and Rasmussen and Ghahramani (2000).

yields the posterior probability of the parameters $\boldsymbol{\theta}$ by updating our prior assumptions on what values they should take with the likelihood of generating the observed data.

While fitting the model amounts to identifying the setting of $\boldsymbol{\theta}$ which maximises the evidence for model $\mathcal{M}_i$ (using say maximum a posteriori inference), the relation to Occam's razor manifests itself in model selection, whereby the most suitable model in a set of candidates $\mathcal{M}$ must be selected. The evidence for a model $\mathcal{M}_i$ (as featured in Equation 1.1) can be evaluated as

$$p\left(\mathcal{D}|\mathcal{M}_i\right) = \int p\left(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i\right) p\left(\boldsymbol{\theta}|\mathcal{M}_i\right) \mathrm{d}\boldsymbol{\theta}, \tag{1.2}$$

which corresponds to the likelihood of the model fitting the data averaged over every possible configuration which $\boldsymbol{\theta}$ may take. A common misconception tied to this correspondence is that Bayesian inference relates to Occam's razor by setting a prior on model parameters. Conversely, as formulated in Equation 1.2, Occam's razor is manifested by way of determining a distribution over datasets, $p\left(\mathcal{D}|\mathcal{M}_i\right)$, that integrates over all values that $\boldsymbol{\theta}$ can take for the chosen model configuration. Assume we are given three models ($\mathcal{M}_1, \mathcal{M}_2$ and $\mathcal{M}_3$) having increasing complexity; by virtue of its higher complexity, $\mathcal{M}_3$ can better explain a wider range of datasets than $\mathcal{M}_1$ or $\mathcal{M}_2$. However, given that the distribution over $\mathcal{D}$ must always integrate to one, this entails that lower probability must be assigned to datasets it cannot adequately explain. This concept is illustrated in Figure 1.1a, which shows how $\mathcal{M}_2$ exhibits the best compromise between confidently fitting the available data and generalisation.

After selecting $\mathcal{M}_2$, let us assume we also identify the parameter setting, $\boldsymbol{\theta}_{\text{OPT}}$, which maximises the posterior probability for the available data; in this case, the marginal likelihood given in Equation 1.2 can be approximated as

$$p\left(\mathcal{D}|\mathcal{M}_2\right) \approx p\left(\mathcal{D}|\boldsymbol{\theta}_{\text{OPT}}, \mathcal{M}_2\right) p\left(\boldsymbol{\theta}_{\text{OPT}}|\mathcal{M}_2\right) \sigma_{\boldsymbol{\theta}|\mathcal{D}}, \tag{1.3}$$

where $\sigma_{\boldsymbol{\theta}|\mathcal{D}}$ denotes the width of the posterior distribution centred at its peak $\boldsymbol{\theta}_{\text{OPT}}$. This is illustrated as $p\left(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_2\right)$ in Figure 1.1b. Under this interpretation, $\sigma_{\boldsymbol{\theta}|\mathcal{D}}$ can be seen as the posterior *uncertainty* associated with the identified optimal parameter configuration. Dividing by the uncertainty, $\sigma_{\boldsymbol{\theta}}$, encoded in the prior distribution over parameters $p\left(\boldsymbol{\theta}|\mathcal{M}_2\right)$ then gives the Occam factor for the given model,

$$p\left(\boldsymbol{\theta}_{\text{OPT}}|\mathcal{M}_2\right) \sigma_{\boldsymbol{\theta}|\mathcal{D}} = \frac{\sigma_{\boldsymbol{\theta}|\mathcal{D}}}{\sigma_{\boldsymbol{\theta}}}. \tag{1.4}$$

This denotes the reduction in uncertainty obtained after fitting the model to observed data. If our a priori assumption is that all settings of $\boldsymbol{\theta}$ are equally likely, multiplying the likelihood term in Equation 1.3 by this ratio thus penalises models which fit the observed data with high likelihood by way of overfitting. As such, Occam's razor arises naturally in Bayesian inference as an implicit means of regularisation which prevents models from becoming too complex, and in turn minimises overfitting in favour of better generalisation.

1.2 Gaussian Processes

In the spirit of Bayesian modelling, Gaussian processes (GPs; Rasmussen and Williams, 2006) define a distribution over candidate functions for modelling observed data. The underlying Bayesian framework allows us to encode prior beliefs on the characteristics of the data being modelled, while also yielding predictions associated by uncertainty estimates indicating the model's confidence in its output. The overarching goal of supervised learning is to construct generalised models for making predictions on previously unseen data. In general, the choice of functions for representing the data is intrinsically restricted by the parameters of the selected model. For example, in the case of linear regression, the range of functions from which the model may be selected are constrained by a predetermined set of basis functions. Similarly, the model capacity of a neural network is largely determined by the selected configuration of hidden layers and neurons. We refer to these models as being *parametric*, whereby the capacity of the model is implicitly linked to the number of parameters available. These assumptions may prove to be overly restrictive in

certain cases where more adaptability is required. GPs achieve greater flexibility by imposing a preference bias as opposed to restrictive constraints; although certain models may be preferred over others, there is no hard restriction on what form the model may take, hence why GPs are categorised as *nonparametric* learning techniques. In this respect, although the parametrisation of GPs allows one to access a certain (infinite) set of functions, preference can be expressed using a prior over candidate functions. Discarding stringent parametric assumptions in favour of placing a prior probability distribution over a family of functions allows for greater freedom in representing data dependencies, thus enabling the construction of better suited models. Just as importantly, this complements the quality of uncertainty estimates expected from such a model.

Consider a supervised learning problem where a set of N input vectors $X = \left[\mathbf{x}_1, \dots, \mathbf{x}_N\right]^\top$ is associated with a set of univariate labels $\mathbf{y} = \left[y_1, \dots, y_N\right]^\top$, where $\mathbf{x}_i \in \mathbb{R}^{D_{\text{in}}}$ and $y_i \in \mathbb{R}$. Formally, GPs are defined as being generalisations of multivariate Gaussian distributions, which can be regarded as probability distributions over functions whereby function values associated with any subset of the input domain have a joint Gaussian distribution. From a generative perspective, observations are modelled via a suitable conditional likelihood $p\left(y_i|f_i\right)$ given latent random variables $\mathbf{f} = \left[f_1, \dots, f_N\right]^\top$, where any subset of $\mathbf{f}$ is assumed to follow a Gaussian distribution. A GP model is fully determined by its mean and covariance, where the specification of a kernel function defines the covariance structure of such random variables,

$$\operatorname{cov}\left(f\left(\mathbf{x}_i\right), f\left(\mathbf{x}_j\right)\right) = k\left(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}\right). \tag{1.5}$$

This kernel function is parametrised by a set of covariance parameters, $\boldsymbol{\theta}$, that determine some of the key characteristics of functions that can be drawn from the GP. A popular choice of covariance is the radial basis, or exponentiated quadratic, function (henceforth referred to as RBF). This is defined as

$$k\left(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}\right) = \sigma^2 \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_j\right)^\top \Lambda^{-1}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right). \tag{1.6}$$

For this kernel, $\boldsymbol{\theta}$ comprises the marginal variance of the GP, σ^2, while $\Lambda = \operatorname{diag}\left[l_1^2, \dots, l_{D_{\text{in}}}^2\right]$ gathers the lengthscale parameters along each dimension of the input domain. This interpretation of the lengthscales allows for automatic relevance determination (ARD; MacKay, 1996; Rasmussen and Williams, 2006), whereby relevant features in the data are weighted by their corresponding lengthscale parameter. This can also be seen as an implicit form of feature selection.

1.2.1 Model Selection

Assimilating to the presentation of Bayes' theorem given in Equation 1.1, the posterior distribution over the parameters $\boldsymbol{\theta}$ can be formulated as

$$p(\boldsymbol{\theta}|X,\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})\,p(\mathbf{f}|X,\boldsymbol{\theta})}{p(\mathbf{y}|X,\boldsymbol{\theta})}, \tag{1.7}$$

where $p(\mathbf{f}|X,\boldsymbol{\theta})$ denotes our prior beliefs on the characteristics of functions that can be drawn from the GP. This is generally encoded in the choice of kernel function and its hyperparameters; for example, the aforementioned RBF kernel is a reasonable choice when the function modelling the data is expected to be smooth, whereas the family of Matérn kernels is better suited when sudden spikes in variance are expected. A comprehensive discussion on kernel selection and design can be found in Duvenaud (2014). Without loss of generality, in this thesis we shall assume the prior mean to be zero.

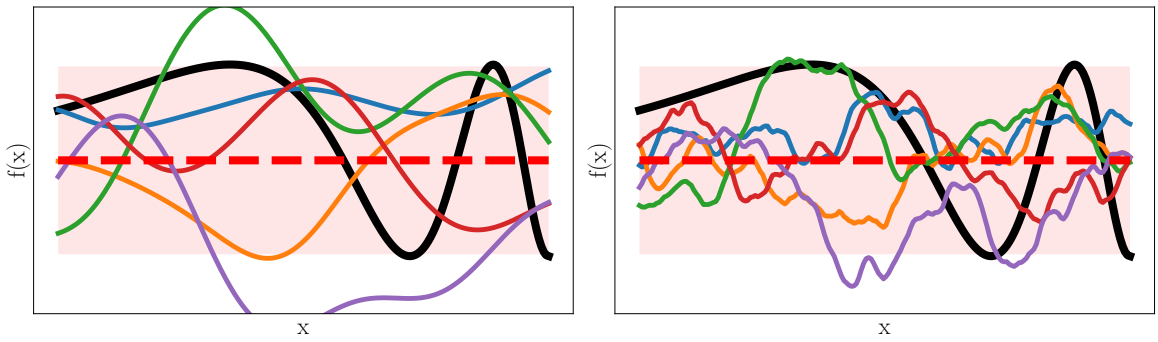
In this setting, the evidence of our model is captured by the marginal likelihood of the GP. Since a GP defines a distribution over candidate functions, this corresponds to averaging over all possible function values that the elements in $\mathbf{f}$ can take. If we assume a Gaussian likelihood with homoscedastic noise variance λ such that $p\left(y_i|f_i\right) = \mathcal{N}\left(y_i|f_i,\lambda\right)$, conjugacy with the prior entails that the marginal likelihood can be analytically evaluated as

$$p(\mathbf{y}|X,\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})\,p(\mathbf{f}|X,\boldsymbol{\theta})\,\mathrm{d}\mathbf{f} = \mathcal{N}\left(\mathbf{0}, K_{\mathrm{XX}} + \lambda I_N\right), \tag{1.8}$$

where K_{XX} is the $N \times N$ symmetric and positive semi-definite matrix evaluated over X with elements $K_{ij} = k\left(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}\right)$, and I_N is the N-dimensional identity matrix. Introducing the notation $K_\lambda = K_{\mathrm{XX}} + \lambda I_N$, the logarithm of the marginal likelihood becomes

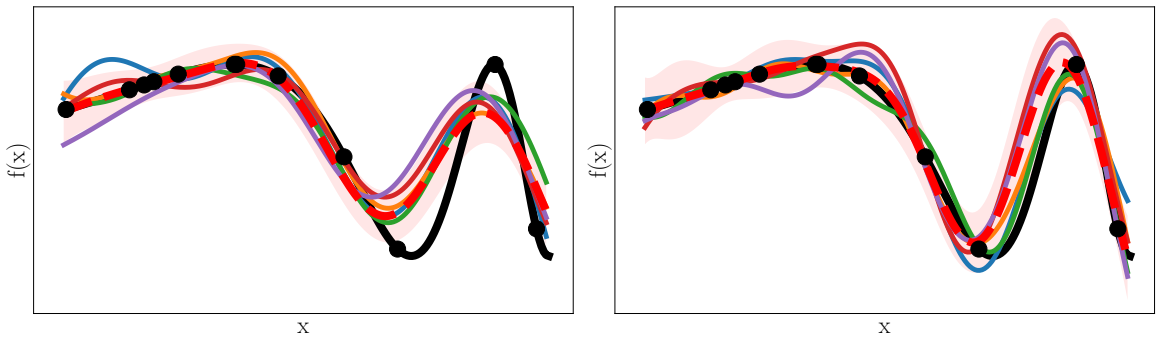
$$\log\left[p(\mathbf{y}|X,\boldsymbol{\theta})\right] = -\frac{1}{2}\log|K_\lambda| - \frac{1}{2}\mathbf{y}^\top K_\lambda^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi. \tag{1.9}$$

The quadratic form appearing in this expression corresponds to the model fit term of the GP, advocating parameter settings that fit the data well. On the other hand, tying in with the principle of Occam's razor highlighted earlier, the log determinant term penalises overly complex models that are characterised by kernel matrices which are diagonally dominant, indicating little interaction between observations. It follows that the optimal parameters $\boldsymbol{\theta}_{\mathrm{OPT}}$ are identified by maximising this objective function using iterative gradient ascent.



(a) Draws from RBF GP prior.

(b) Draws from Matérn 3/2 GP prior.



(c) Draws from GP posterior with initial $\boldsymbol{\theta}$.

(d) Draws from GP posterior with $\boldsymbol{\theta}_{\text{OPT}}$.

Fig. 1.2 Illustration of Gaussian process inference. The true function is coloured in black, while markers denote observed data-points. Dashed red lines indicate the mean prediction while shaded regions depict the 95% confidence interval.

1.2.2 Predictions

Given that GPs capture distributions over candidate functions, it follows that predictions obtained from the model can also be a described by a multivariate Gaussian distribution characterised by its mean and variance, where the latter can be interpreted as the model's uncertainty on the prediction. Using standard Gaussian identities, and for a given setting of $\boldsymbol{\theta}$, the predictive distribution for a test point $\mathbf{x}_\star$ can be evaluated as follows,

$$p\left(f_\star|\mathbf{y}, X, \mathbf{x}_\star, \boldsymbol{\theta}\right) = \int p\left(f_\star|\mathbf{f}, X, \mathbf{x}_\star, \boldsymbol{\theta}\right) p\left(\mathbf{f}|\mathbf{y}, X, \boldsymbol{\theta}\right) \mathrm{d}\mathbf{f} = \mathcal{N}\left(f_\star|\mu_\star, \Sigma_\star\right), \tag{1.10}$$

where

$$\mu_\star = \mathbf{k}_\star^\top K_\lambda^{-1}\mathbf{y}, \quad \text{and} \tag{1.11}$$

$$\Sigma_\star = k_{\star\star} - \mathbf{k}_\star^\top K_\lambda^{-1}\mathbf{k}_\star. \tag{1.12}$$

In the equations above, $\mathbf{k}_\star$ corresponds to the covariance between $\mathbf{x}_\star$ and observed training data, while $k_{\star\star}$ denotes the covariance of $\mathbf{x}_\star$ with itself. Once again assuming a Gaussian likelihood with noise variance λ, the prediction $y_\star$ is given by

$$y_\star \sim \mathcal{N}\left(\mu_\star, \Sigma_\star + \lambda\right). \tag{1.13}$$

1.2.3 Illustrative Example

For clarity, we provide a simple example for visualising how inference with GPs works in practice. Before observing any data, the GP prior encodes our beliefs on the properties of functions we expect to appropriately fit the data. In Figures 1.2a and 1.2b, we illustrate functions drawn from two distinct priors, one which is relatively smooth and another which captures more oscillatory behaviour. While both are indeed valid priors, we select the first prior as it better reflects the target function's behaviour.

By way of Bayesian inference, this prior over functions is then updated using the likelihood component of our model. The effect of conditioning on observed data-points is illustrated in Figure 1.2c, which now features functions drawn from the posterior distribution of the GP. Nonetheless, the fit obtained using the initial set of parameters $\boldsymbol{\theta}$ may not be the best fit attainable by the model. As evidenced by the superior function draws illustrated in Figure 1.2d, maximising the marginal likelihood of the model yields the optimum configuration of hyperparameters $\boldsymbol{\theta}_{\text{OPT}}$.

1.2.4 Non-Gaussian Likelihoods

When the likelihood $p\left(y_i|f_i\right)$ is not Gaussian, such as in classification problems, it is no longer possible to analytically integrate out latent variables. Instead, techniques such as Gaussian approximations (Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008), and methods attempting to characterise the full posterior $p\left(\mathbf{f}, \boldsymbol{\theta}|\mathbf{y}\right)$ (Filippone et al., 2013; Murray et al., 2010) may be required. In order to preserve focus on the methodology being presented, and unless stated otherwise, the models discussed in this manuscript will initially be framed in the context of regression problems where observations are assigned a Gaussian likelihood. Extensions to more involved likelihoods and classification problems are nevertheless provided for both of the primary contributions covered in this thesis.

1.3 Gaussian Processes in the Big Data Regime

The success of nonparametric models built around kernels hinges on the adaptation and optimisation of $\boldsymbol{\theta}$. However, the scalability of these models is predominantly hindered by linear algebraic operations having large computational complexity. On inspection of the GP marginal likelihood given in Equation 1.9, we can observe that evaluating this expression involves the computation of the log determinant of K_λ, as well as a quadratic term involving this same matrix, both of which have time and space complexity of $\mathcal{O}\left(N^3\right)$ and $\mathcal{O}\left(N^2\right)$. Computing the log marginal likelihood can be bypassed if we only carry out gradient-based optimisation. In the regression case, the gradients can be computed as

$$\frac{\partial \log\left[p\left(\mathbf{y}|X,\boldsymbol{\theta}\right)\right]}{\partial \theta_i} = -\frac{1}{2}\mathrm{Tr}\left(K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}K_\lambda^{-1}\mathbf{y}. \tag{1.14}$$

Nonetheless, although we indeed avoid computing the log determinant, we are still required to evaluate other terms involving the inversion of K_λ. Factorising the kernel matrix K_λ into $LL^\top$ (where L is a triangular matrix) using the Cholesky decomposition necessitates $\mathcal{O}\left(N^3\right)$ operations. The trace term in the calculation of the gradient also requires $\mathcal{O}\left(N^3\right)$ operations, and similar computations are required for computing mean and variance predictions for test data. As discussed, the likelihood mapping latent function values to observations may not always be Gaussian either, as will be the case for classification problems. Under these conditions, inference is no longer analytic and further approximations must be introduced.

In view of these constraints, GP inference is often too expensive, or even intractable, when the size of the data exceeds just a few thousand points. Consequently, devising approaches for enabling scalable GP inference without compromising on accuracy is a recurring theme in the literature. Such approximations were rigorously explored by Quinonero-Candela and Rasmussen (2005), Snelson and Ghahramani (2007) and Titsias (2009) among many others, and these developments were recently summarised by Liu et al. (2018). Developing approaches for improving the tractability of GPs while still preserving good predictive performance serves as the primary motivation for the work undertaken in this thesis. In particular, the first contribution presented here is posited as a principled approach for carrying out exact inference on a computational budget without resorting to traditional model approximations. Contrary to standard techniques for scaling GPs, the proposed methodology achieves scalability by accelerating the evaluation of computationally expensive algebraic operations using unbiased approximations. More specifically, we

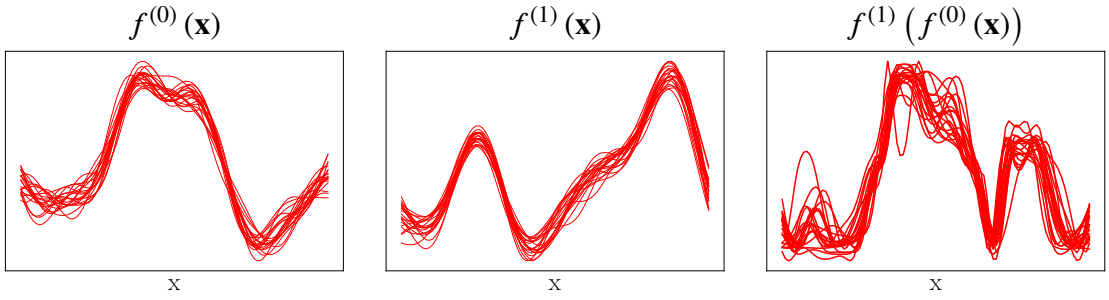


Fig. 1.3 Visualisation of stochastic process composition in a deep Gaussian process.

develop a suite of preconditioners for kernel matrices which can be combined with stochastic trace estimation for accelerating computation in both GP regression and classification tasks without restricting the overall capacity of the model. This is particularly pertinent to applications such as medical diagnosis where high precision and well-calibrated uncertainty is essential.

An investigation into the numerical uncertainty introduced by such schemes is also featured in this thesis, leveraging advances in probabilistic numerics (Hennig et al., 2015) to give a novel perspective on the sources of uncertainty that must be taken into consideration when working with budget-constrained evidence. We contribute towards the literature by proposing a probabilistic scheme for estimating the log determinant of large matrices under a Bayesian framework, and give an exploratory analysis of how this can be combined with probabilistic linear solvers in order to quantify the uncertainty introduced by approximating linear algebra in the context of large-scale kernel methods.

1.4 Gaussian Processes in the Deep Learning Landscape

The resilient appeal of Bayesian inference schemes in the advent of competitive deep learning techniques (Goodfellow et al., 2016; LeCun et al., 2015) can be attributed to their well-founded quantification of uncertainty. Bridging the gap between GPs and widely-used deep learning models remains a persistent research goal which also features heavily in this work. There has been sustained interest in addressing this goal, and contributions have varied from designing more sophisticated kernels to mimic the behaviour of neural network architectures (Cho and Saul, 2009; Mairal et al., 2014) to optimising the GP model with respect to objective functions alternative to the marginal likelihood (Sundararajan and Keerthi, 2001). In this thesis, we outline an approach for jointly exploring these directions along with developing superior variational approximations to the true posterior

distribution of the GP. In doing so, we obtain state-of-the-art results for GPs on several deep learning benchmarks, and provide a new perspective on the range of problems that can be targeted using GPs.

Appealing to the structure of deep neural networks, the composition of multiple GPs as a deep Gaussian process (DGP; Damianou and Lawrence, 2013) enables a deep probabilistic nonparametric approach to flexibly tackle complex machine learning problems while also providing well-calibrated uncertainty quantification. An illustrative example of how process composition is suitable for modelling complex functions is given in Figure 1.3. However, although DGPs appear to be best positioned to match the results obtained by modern neural network architectures, existing inference approaches for DGP models have limited scalability and are notoriously cumbersome to construct (Bui et al., 2016; Dai et al., 2016). To this end, another significant contribution of this thesis is the development of a novel, scalable formulation of DGPs based on random feature expansions that can handle large-scale problems far beyond the scale of datasets to which GPs, and especially DGPs, are typically applied. Such a contribution is of great importance for widening the appeal of DGPs to both researchers in the broader machine learning community and practitioners alike.

1.5 Outline and Contributions of Thesis

The content of this thesis is organised as follows:

- In **Chapter 2**, we investigate state-of-the-art techniques for scaling Gaussian processes to big datasets, covering three main categories of approximations, namely inducing point-based, spectral, and structure exploiting approximations. This chapter is intended to equip the reader with the background knowledge required for apprehending the underlying concepts presented in this thesis, and clarify how our contributions fit within the landscape of existing research on scalable Gaussian process inference;

- **Chapter 3** covers the first primary contribution of this thesis, where we present a thorough investigation of how carefully selected preconditioners may be incorporated within a scalable approach for both solving kernel machines and learning their hyperparameters. This allows for exact Gaussian process inference on a computational budget, which outperforms standard sparse approximations;

- The methodology presented in Chapter 3 relies on unbiased approximations to linear algebra for accelerating inference. However, this introduces an additional degree of computational (or numerical) uncertainty within the model which is generally ignored due to the difficulty of properly quantifying it. Inspired by recent advances in the field of probabilistic numerics, in **Chapter 4** we propose a novel Bayesian interpretation for estimating the log determinant of a matrix. Coupling this contribution with contemporaneous work on probabilistic linear solvers, we also present a preliminary discussion on how to combine these concepts in Gaussian process inference, and provide a cautionary note on how, in spite of their theoretic appeal, such constructions may not yet be robust enough to meet this goal;

- **Chapter 5** starts by briefly describing AutoGP - a model we developed with the primary intent of exploring the capabilities and limitations of Gaussian process models in relation to deep learning techniques. This segues into a discussion on how deep Gaussian processes are a more natural candidate for matching the performance of neural networks. In a departure from the shallow Gaussian process models discussed in the preceding chapters, we then develop a novel formulation of deep Gaussian processes built upon random feature expansions, and demonstrate how such a model leads to considerable performance improvements over pre-existing deep Gaussian process models;

- Finally, in **Chapter 6**, we summarise the contributions presented in this thesis and give an insightful retrospective on the breakneck rate at which this field of study is progressing. We conclude the thesis by tying this discussion to an outlook on possible extensions and future work.

Chapter 2

Scalable Gaussian Process Inference

Developing scalable learning models without compromising performance is at the forefront of machine learning research. As highlighted in the introduction to this thesis, the scalability of Gaussian processes is predominantly hindered by linear algebraic operations with cubic computational complexity, which deters their application to datasets having more than a few thousand observations. In this chapter, we review the literature on developing approximations for scalable Gaussian process inference with particular emphasis on approaches that will feature in subsequent chapters. Although the development of scalable deep Gaussian processes is also a major contribution of this thesis, we defer the introduction and discussion of these models to Chapter 5 in order to streamline the presentation of different models.

2.1 Overview

The success of deep learning in tasks such as image classification, speech recognition, and other tasks emulating human expertise have shifted the onus to gathering larger datasets for effectively training such models (LeCun et al., 2015). In contrast, Gaussian processes (henceforth GPs) are predominantly regarded as data-efficient models which are capable of yielding sensible predictions even when only few observations are available. Moreover, their extension to large datasets is heavily burdened by the computational and storage complexity associated with their formulation, $\mathcal{O}\left(N^3\right)$ and $\mathcal{O}\left(N^2\right)$ respectively. To this end, a plethora of approaches have been investigated and developed for circumventing these constraints with the ultimate goal of applying GPs to larger datasets. This chapter is

divided into three main sections that each describe a broad category of GP approximations. Although similar literature reviews, most recently carried out by Liu et al. (2018), have considered alternative groupings, in this text we opt for the following split:

- **Inducing Point Approximations**: Arguably the most prevalent in the literature, a variety of approaches based on inducing points have targeted scalability by way of approximating either the prior or posterior of the GP model using low-rank formulations of the central $N \times N$ kernel matrix. We start by describing foundational work carried out by Csató and Opper (2002), Snelson and Ghahramani (2005), and Quinonero-Candela and Rasmussen (2005), before progressing to the more principled variational free energy approximation put forward by Titsias (2009) and championed by Matthews et al. (2016). A selection of these methods reappear in Chapter 3, where they are reinterpreted as preconditioners for enabling tractable exact GP inference. An introduction to stochastic variational inference for GPs (Hensman et al., 2013) is also given here;

- **Random Feature Approximations**: Assimilating to the weight-space view of a GP, random feature approximations rely on the spectral representation of covariance functions (Rahimi and Recht, 2008) for scalable kernel learning. These approaches have markedly different properties from the aforementioned inducing point methods, and are sometimes criticised for relaxing the true nonparametric form of GP modelling. In this section, we focus on the sparse spectrum GP (Lázaro-Gredilla et al., 2010) and its extension (Gal and Turner, 2015), concluding with a brief overview of the more recently introduced variational Fourier features (Hensman et al., 2017). The deep Gaussian process approximation we propose in Chapter 5 leverages random feature expansions of kernels for accelerating inference;

- **Structure Exploiting Approximations**: This final category covers techniques that exploit problem structure for simplifying computationally expensive linear algebra. Extending the use of Kronecker products for accelerating GP inference when grid-structured inputs are available, structured kernel interpolation (Wilson and Nickisch, 2015) has became a staple GP approximation which has been extended in several complementary directions (Dong et al., 2017; Evans and Nair, 2018; Pleiss et al., 2018). Structured kernel interpolation also appears as a preconditioner in the next chapter, while results obtained by deep kernel learning combined with structured kernel interpolation (Wilson et al., 2016b) are cited as baselines for the methods developed in Chapter 5.

2.2 Inducing Point Approximations

An immediate approach for reducing the computational complexity of GP inference and hyperparameter optimisation involves sub-sampling the available training data and discarding points beyond a chosen threshold. By reducing the size of the training data to $M \ll N$, the computational complexity is also lowered to $\mathcal{O}\left(M^3\right)$. Nevertheless, while effective in its crude simplicity, this approach (known as subset of data) forsakes potentially important observations in the full training set under the assumption of overall redundancy, resulting in a model that is likely to be erroneously overconfident in its predictions.

2.2.1 Traditional Approaches

On the other hand, we have an innate preference for methods that do not discard training data upfront, but rather rely on low-rank decompositions of the full kernel matrix in order to accelerate computation. Recalling that $K_\lambda = K_{\mathrm{XX}} + \lambda I_N$, we would be interested in obtaining a rank-M decomposition such that $K_\lambda \approx A_{\mathrm{NM}} B_{\mathrm{MM}} C_{\mathrm{MN}} + \lambda I_N$. Such structure enables application of the Woodbury inversion lemma for inverting K_λ,

$$K_\lambda^{-1} \approx \left(A_{\mathrm{NM}} B_{\mathrm{MM}} C_{\mathrm{MN}} + \lambda I_N\right)^{-1} = \frac{1}{\lambda} I_N - \frac{1}{\lambda} A_{\mathrm{NM}} \left(\lambda B_{\mathrm{MM}}^{-1} + C_{\mathrm{MN}} A_{\mathrm{NM}}\right)^{-1} C_{\mathrm{MN}}. \quad (2.1)$$

Bypassing the Cholesky decomposition for inverting K_λ sharply reduces the computational complexity from $\mathcal{O}\left(N^3\right)$ to $\mathcal{O}\left(M^2 N\right)$. Similarly, the approximated log determinant can be evaluated as

$$\log |K_\lambda| \approx \log |B_{\mathrm{MM}}| + \log |\lambda B_{\mathrm{MM}}^{-1} + C_{\mathrm{MN}} A_{\mathrm{NM}}|. \quad (2.2)$$

The most faithful low-rank decomposition of the kernel matrix is given by its eigendecomposition into M eigenvectors, Q_{NM}, and eigenvalues, Λ_{MM}. This corresponds to $K_{\mathrm{XX}} \approx Q_{\mathrm{NM}} \Lambda_{\mathrm{MM}} Q_{\mathrm{NM}}^\top$. However, the complexity of computing the eigendecomposition itself is also $\mathcal{O}\left(N^3\right)$, resulting in no computational savings. This exact decomposition can be avoided by considering the Nyström approximation (Williams et al., 2002; Williams and Seeger, 2001) instead,

$$K_{\mathrm{XX}} \approx K_{\mathrm{XU}} K_{\mathrm{UU}}^{-1} K_{\mathrm{XU}}^\top. \quad (2.3)$$

In the above, K_{UU} is computed over a subset $M \ll N$ of the training data, which is referred to as the active set. As shown by Snelson and Ghahramani (2005), this subset of points does not have to be sampled directly from the original training data as long as they reside in the same input space, which entails that their location can be optimised during training. These can thus be referred to as pseudo-inputs, but in this thesis we opt for the more generally used *inducing points* to cover both options.

This approximation forms the basis of the fundamental subset of regressors (SOR; Silverman, 1985; Smola and Bartlett, 2001) and deterministic training conditional (DTC; Csató and Opper, 2002; Seeger et al., 2003) approximations. Due to their similarity, we shall primarily focus on DTC, which improves upon the former by ensuring that the test conditional remains exact and only the training conditional is approximated. This guarantees that the model variance is less likely to be underestimated than with SOR. Although the initial presentation of DTC precedes the unifying framework developed by Quinonero-Candela and Rasmussen (2005), we follow the notation of the latter in order to simplify exposition. Denoting the inducing variables evaluated at the selected input locations as $\mathbf{u}$, it is initially assumed that $\mathbf{f}$ and $\mathbf{f}_\star$ are conditionally independent given $\mathbf{u}$,

$$p\left(\mathbf{f}, \mathbf{f}_\star\right) \approx q\left(\mathbf{f}, \mathbf{f}_\star\right) = \int q\left(\mathbf{f}_\star|\mathbf{u}\right) q\left(\mathbf{f}|\mathbf{u}\right) p\left(\mathbf{u}\right) \mathrm{d}\mathbf{u}, \tag{2.4}$$

where $p(\mathbf{u}) = \mathcal{N}\left(\mathbf{0}, K_{\mathrm{UU}}\right)$. Sparsity is then induced by considering an approximation $q\left(\mathbf{f}|\mathbf{u}\right)$ to the conditional $p\left(\mathbf{f}|\mathbf{u}\right)$. As its name implies, the DTC approach treats the M inducing points as being deterministic, hence the training conditional is

$$q_{\mathrm{DTC}}\left(\mathbf{f}|\mathbf{u}\right) = \mathcal{N}\left(\mathbf{f}|K_{\mathrm{XU}}K_{\mathrm{UU}}^{-1}\mathbf{u}, \mathbf{0}\right). \tag{2.5}$$

This allows for the inducing points to be immediately marginalised out of the posterior distribution, yielding $q_{\mathrm{DTC}}\left(\mathbf{y}|\mathbf{u}\right) = \mathcal{N}\left(\mathbf{y}|K_{\mathrm{XU}}K_{\mathrm{UU}}^{-1}\mathbf{u}, \lambda I_N\right)$. Efficient training and inference is then achieved by application of the algebraic identities presented in Equations 2.1 and 2.2. Although this is already a marked improvement over the subset of data approach, treating inducing points deterministically negatively constrains the model's flexibility. Furthermore, the distinction between latent function values for training and test points is also unnatural for GP modelling.

The fully independent training conditional (FITC; Snelson and Ghahramani, 2005) builds upon the assumption of independence across $\mathbf{f}$ developed in the DTC approach by introducing a correction to the prior such that the diagonal elements of the covariance matrix remain exact,

$$p(\mathbf{f}) = \mathcal{N}\left(\mathbf{f}|\mathbf{0}, \widetilde{K}_{\mathrm{XX}} + \mathrm{diag}\left[K_{\mathrm{XX}} - \widetilde{K}_{\mathrm{XX}}\right]\right), \tag{2.6}$$

where $\widetilde{K}_{\mathrm{XX}}$ denotes the Nyström approximation described in Equation 2.3. The training conditional can thus be updated as

$$q_{\mathrm{FITC}}(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^{N} p\left(f_i|\mathbf{u}\right) = \mathcal{N}\left(\mathbf{f}|K_{\mathrm{XU}}K_{\mathrm{UU}}^{-1}\mathbf{u}, \mathrm{diag}\left[K_{\mathrm{XX}} - \widetilde{K}_{\mathrm{XX}}\right]\right). \tag{2.7}$$

Extending this concept, the partially independent training conditional (PITC; Quinonero-Candela and Rasmussen, 2005) approximation corrects blocks along the diagonal, introducing some degree of partitioning in the data. However, the performance improvement of using PITC over FITC is minor, particularly when the basis functions are local as for the RBF kernel. Furthermore, an additional pre-processing step is required for clustering observations into suitable partitions (Snelson and Ghahramani, 2007).

2.2.2 Variational Free Energy Approximation

The approaches considered thus far all rely on approximating the GP prior by some approximation $q(\mathbf{f}|\mathbf{u})$. Under this setting, the inducing points take the form of kernel hyperparameters to be additionally optimised along with the regular parameters. However, this treatment is very likely to result in overfitting. More significantly, by altering the prior we are no longer approximating the exact GP model; on the contrary, the model takes on a different form altogether. A widely-used approach for enabling tractability in Bayesian inference involves the use of variational inference techniques (Blei et al., 2017; Jordan et al., 1999; Zhang et al., 2017). By way of optimisation, the goal of variational inference is to identify the approximate posterior in a predefined family of probability densities that most closely matches the true, but computationally intractable, posterior.

The variational free energy (VFE) approximation given by Titsias (2009) targets this issue by exploiting variational inference in order to approximate the posterior distribution itself. In particular, the marginal likelihood over $\mathbf{y}$ is approximated with $q(\mathbf{y})$, and taking this to be a variational approximation recasts our optimisation problem as minimising the Kullback-Leibler divergence (D_{KL}; Gray, 1990) between the true and approximate posterior,

$$\underset{q(\mathbf{f},\mathbf{u})}{\arg\min}\, D_{\mathrm{KL}}\left[q(\mathbf{f},\mathbf{u})\,||\,p(\mathbf{f},\mathbf{u}|\mathbf{y})\right]. \tag{2.8}$$

For preserving computational efficiency, $q(\mathbf{f}, \mathbf{u})$ is assumed to factorise as $p(\mathbf{f}|\mathbf{u})\, q(\mathbf{u})$. Using Jenson's inequality, a variational lower bound can then be obtained on the true log marginal likelihood as follows,

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|\mathbf{u})\, p(\mathbf{u})\, \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \log \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|\mathbf{u})\, p(\mathbf{u}) \times \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&\geq \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})\, \cancel{p(\mathbf{f}|\mathbf{u})}\, p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})}\, q(\mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{u})}\left[p(\mathbf{y}|\mathbf{f})\right]}_{\text{model fit}} - \underbrace{D_{\mathrm{KL}}\left[q(\mathbf{u})\,||\,p(\mathbf{u})\right]}_{\text{regularisation}}.
\end{aligned} \tag{2.9}$$

This is referred to as the *evidence lower bound* on the marginal likelihood, in which the first term favours models that properly fit the data, while the subtracted term penalises models that deviate too far from the prior. One of the most appealing properties of this formulation is that when all training observations are used as inducing points, the exact GP model is recovered. This happens because the D_{KL} term given in Equation 2.8 collapses to zero when the variational posterior matches the true formulation.

The fundamental differences between FITC and VFE were succinctly detailed and explored by Bauer et al. (2016). Among other properties, the authors highlight that whereas FITC is heavily prone to underestimating the noise variance λ and does not guarantee superior fits when more inducing points are made available, VFE consistently identifies good solutions while model fit improves when more inducing points are allocated. On the downside, however, variational approximations are also more susceptible to get stuck in local optima, and sensible optimisation strategies must be designed to ensure convergence to the global optimum. A further theoretical investigation into the connection between the described variational inducing point framework and divergence measures for stochastic processes was carried out by Matthews et al. (2016), validating the correctness of the original proposal in Titsias (2009). Finally, recent work by Bui et al. (2017) presents a unifying view of variational approximations under a novel power expectation propagation framework.

2.2.3 Stochastic Variational Inference

In spite of the favourable reduction in computational and storage complexity obtained using either of the aforementioned approximations, $\mathcal{O}\left(M^2 N\right)$ complexity would still be unfeasible when the available data runs into hundreds of thousands or even millions of observations. In view of this requirement, Hensman et al. (2013) leverage the remarkable success obtained by stochastic optimisation techniques on training deep neural networks by developing a stochastic variational inference (SVI) framework for GPs. This is reminiscent of the work carried out by Hoffman et al. (2013), in which SVI techniques were used to approximate posterior distributions for a suite of probabilistic models. This concept will be particularly pertinent to the models discussed in Chapter 5, where the focus is shifted to scaling GPs and their deep counterparts to truly large datasets in the order of millions of observations.

At its core, SVI for GPs borrows from the stochastic optimisation techniques employed for training neural network architectures by enabling the use of mini-batch-based inference; at every iteration of the optimisation procedure, a stochastic gradient is computed using a single mini-batch rather than the full dataset. Given that the mini-batch size can be set to be as small as one, the overall complexity is now independent of the number of training points N, and can be reduced to $\mathcal{O}\left(M^3\right)$ per iteration. In VFE, it was possible to collapse $q(\mathbf{u})$ by deriving the optimal variational distribution analytically. Such analytic tractability is no longer preserved if we only have access to a single mini-batch of the full data at any given time, and we are instead required to keep an explicit global representation of the variational approximation of the inducing points, i.e. $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote mean and covariance parameters to be optimised. If the likelihood is additionally assumed to be fully-factorised over the available observations, the lower bound on the marginal likelihood becomes

$$\mathcal{L}_{\mathrm{VB}} = \frac{N}{|\mathcal{I}_{\mathrm{MB}}|} \sum_{y_i \in \mathcal{I}_{\mathrm{MB}}} \log p\left(y_i|f_i\right) p\left(f_i|\mathbf{u}\right) q(\mathbf{u}) - D_{\mathrm{KL}}\left[q(\mathbf{u})\,||p(\mathbf{u})\right], \tag{2.10}$$

where the mini-batch consists of training observations indexed by $\mathcal{I}_{\mathrm{MB}}$. While not essential to the proposed technique, incorporating natural gradients in the optimisation procedure was shown to yield quicker convergence to an optimal solution. This was also recently corroborated in work by Salimbeni et al. (2018).

2.3 Random Features Approximations

The second category of GP approximations considered in this chapter relies on exploiting the dual interpretation of a kernel by way of random feature maps. The concepts discussed here are a prerequisite to the material presented later in Chapter 5, where we propose a novel deep Gaussian process model inspired by the approximations featured in this section.

2.3.1 Spectral Representation of Kernels

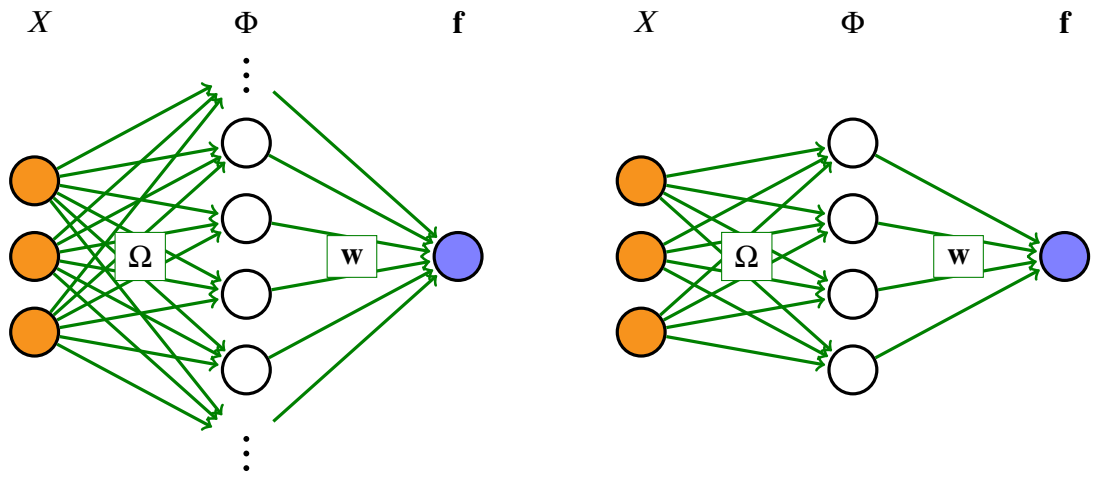
In demonstrating how inner products of random features can effectively approximate a wide range of kernels, the seminal work by Rahimi and Recht (2008) is unanimously considered to be one of the most influential papers published in the previous decade, having had significant impact on research communities working on kernel-based models such as support vector machines, kernel ridge regression, and ultimately GPs. Their work is primarily motivated by Bochner's theorem (Rudin, 1990), which states that any continuous shift-invariant normalised covariance function $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = k\left(\mathbf{x}_i - \mathbf{x}_j\right)$ is said to be positive definite if and only if it can be rewritten as the Fourier transform of some non-negative measure.

Adapting this theorem to different covariance functions largely depends on the choice of spectral measure. Denoting spectral frequencies by $\boldsymbol{\omega}$, while assigning $\imath = \sqrt{-1}$ and a distance measure $\boldsymbol{\delta} = \mathbf{x}_i - \mathbf{x}_j$, the RBF covariance listed in Equation 1.6 can be represented in the Fourier space as

$$k\left(\mathbf{x}_i, \mathbf{x}_j | \boldsymbol{\theta}\right) = \sigma^2 \int p\left(\boldsymbol{\omega}\right) e^{\imath \boldsymbol{\delta}^\top \boldsymbol{\omega}} \mathrm{d}\boldsymbol{\omega} = \sigma^2 \mathbb{E}_{p(\boldsymbol{\omega})}\left[e^{\imath \boldsymbol{\delta}^\top \boldsymbol{\omega}}\right], \tag{2.11}$$

with a corresponding non-negative measure $p(\boldsymbol{\omega}) = \mathcal{N}\left(\mathbf{0}, \Lambda^{-1}\right)$ encoding the lengthscale parameters. By definition, the RBF covariance will always yield non-negative real values; therefore, the imaginary component can be dropped from the expectation appearing in Equation 2.11, which is then simplified to $\mathbb{E}_{p(\boldsymbol{\omega})}\left[\cos\left(\boldsymbol{\delta}^\top \boldsymbol{\omega}\right)\right]$. The covariance can thus be approximated by sampling N_{RF} spectral features from $p\left(\boldsymbol{\omega}\right)$ using a Monte Carlo (henceforth MC) procedure, such that

$$k\left(\mathbf{x}_i, \mathbf{x}_j | \boldsymbol{\theta}\right) \approx \frac{\sigma^2}{N_{\mathrm{RF}}} \sum_{r=1}^{N_{\mathrm{RF}}} \cos\left(\delta^\top \widetilde{\boldsymbol{\omega}}_r\right), \tag{2.12}$$



(a) In the limit of infinitely many activation units, a single-layered neural network can be made equivalent to a GP.

(b) In the practical setting having a finite number of activations, a sparse GP approximation is obtained.

Fig. 2.1 Graphical illustration of the sparse spectrum Gaussian process. The spectral decomposition of a kernel using random feature maps can be leveraged to develop a sparse GP model in the weight-space view with lower computational complexity.

where $\widetilde{\boldsymbol{\omega}}$ are samples drawn from $p(\boldsymbol{\omega})$. In order to recover the inner product representation of the kernel, i.e. $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \boldsymbol{\phi}\left(\mathbf{x}_i\right)^{\top} \boldsymbol{\phi}\left(\mathbf{x}_j\right)$, we then set

$$\boldsymbol{\phi}(\mathbf{x}) = \left[\cos\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_1\right) \sin\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_1\right), \ldots, \cos\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_{N_{\mathrm{RF}}}\right) \sin\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_{N_{\mathrm{RF}}}\right)\right]^{\top}. \tag{2.13}$$

Defining $\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = \left[\cos\left(\mathbf{x}^{\top}\boldsymbol{\omega}\right), \sin\left(\mathbf{x}^{\top}\boldsymbol{\omega}\right)\right]^{\top}$, the MC estimation of the covariance can hence be summarised as

$$k\left(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}\right) \approx \frac{\sigma^2}{N_{\mathrm{RF}}} \sum_{r=1}^{N_{\mathrm{RF}}} \mathbf{z}\left(\mathbf{x}_i|\widetilde{\boldsymbol{\omega}}_r\right)^{\top} \mathbf{z}\left(\mathbf{x}_j|\widetilde{\boldsymbol{\omega}}_r\right). \tag{2.14}$$

This has an important practical implication as it provides the means to access an approximate explicit representation of the mapping induced by the covariance function that in the RBF case is known to be infinite dimensional (Shawe-Taylor and Cristianini, 2004).

2.3.2 Sparse Spectrum Gaussian Processes

In our presentation of GPs thus far, we have adhered to the so-called *function-space* view of the model. However, there exists a complementary view that explicitly defines a GP as a weighted sum of nonlinear basis functions, informally referred to as the *weight-space* view (Rasmussen and Williams, 2006). This interpretation is rooted in the analysis of Neal (1995), who showed that under certain conditions, a one-layered neural network with infinitely many activation units is equivalent to a GP. This implies the following representation,

$$f(\mathbf{x}) = \frac{1}{N_{\mathrm{H}}} \mathbf{h}(\mathbf{x})^{\top} \mathbf{w}, \tag{2.15}$$

whereby basis functions obtained by nonlinear transformations of $\mathbf{x}$ using

$$\mathbf{h}(\mathbf{x}) = \left[h_1(\mathbf{x}), \ldots, h_{N_{\mathrm{H}}}(\mathbf{x}) \right]^{\top} \tag{2.16}$$

are linearly combined by a column vector of weights $\mathbf{w}$. By invoking the central limit theorem, it can be shown that as the number of activation units N_{H} tends towards infinity, the joint distribution between any function values obtained from this model increasingly resembles a Gaussian. A graphical illustration of this architecture is given in Figure 2.1, which depicts both a neural network with infinite activation units corresponding to an exact GP (Figure 2.1a), and a more practical GP approximation derived from considering a finite number of activations (Figure 2.1b). For the time being, the random feature map Ω can be assumed to be an identity mapping; however, as will be discussed further below, this has to be chosen more carefully in practice.

This perspective ties in nicely with the kernel approximation given in Equation 2.14 because a suitable spectral representation of a designated kernel can be employed to carry out GP inference in the weight-space view. As originally presented in Lázaro-Gredilla et al. (2010), sparsity in a sparse spectrum GP (SSGP) is achieved by setting Ω and the activation function $h(\cdot)$ in such a way as to approximate the behaviour of a stationary kernel. In particular, transforming X using the random feature map Ω will yield a set of *random features*, $X\Omega$, that become basis functions by application of $\mathbf{h}(\cdot)$. This makes it possible to reformulate the original inference problem with an explicit finite representation given by the expansion using random features.

For the RBF covariance, these are additionally constrained to be Fourier features. Defining $\Omega = \left[\widetilde{\boldsymbol{\omega}}_1, \ldots, \widetilde{\boldsymbol{\omega}}_{N_{\text{RF}}}\right]$, this representation requires the input X to be transformed into a new design matrix as follows,

$$\Phi = \left[\sqrt{\frac{\sigma^2}{N_{\text{RF}}}}\cos\left(X\Omega\right), \sqrt{\frac{\sigma^2}{N_{\text{RF}}}}\sin\left(X\Omega\right)\right], \tag{2.17}$$

where sine and cosine operations are applied element-wise to their argument. In this manner, the original GP model is now 'linearised' and can be treated as a standard Bayesian linear model having design matrix Φ. If we consider a set of weights in vector form $\mathbf{w}$, the latent functions can therefore be expressed as $\mathbf{f} = \Phi\mathbf{w}$. The values of $\mathbf{w}$ denote the coefficients of linear combinations of the random Fourier features to obtain the latent functions modelling the output labels. As before, we limit the discussion in this section to the univariate output case, but this can easily be extended to multi-output problems by considering a matrix of weights $W = \left[\mathbf{w}_1, \ldots, \mathbf{w}_{D_{\text{out}}}\right]$ instead. Analysing the mean and covariance of a given latent function when any individual weight w_i is sampled from $\mathcal{N}\left(w_i|0,1\right)$, it can easily be verified that the prior mean of the latent functions is zero and $\Phi\Phi^\top \approx K_{\text{XX}}$,

$$\mathbb{E}\left[\mathbf{f}\right] = \mathbb{E}\left[\Phi\mathbf{w}\right] = \Phi\mathbb{E}\left[\mathbf{w}\right] = \mathbf{0}, \quad \text{and} \tag{2.18}$$

$$\begin{aligned}\text{cov}[\mathbf{f}] &= \mathbb{E}\left[\Phi\mathbf{w}\mathbf{w}^\top\Phi^\top\right] = \Phi\mathbb{E}\left[\mathbf{w}\mathbf{w}^\top\right]\Phi^\top \\ &= \Phi\Phi^\top \approx K_{\text{XX}}.\end{aligned} \tag{2.19}$$

Although $\mathbf{w}$ can generally be sampled from any distribution, the correspondence between a single-layered neural network with a finite number of activation units and an approximate GP only holds if these are assumed to follow a Gaussian distribution. If we limit our discussion to GP regression problems with a Gaussian likelihood, the posterior mean over $\mathbf{w}$ turns out to be the standard regularised least squares estimator. As a result, it is possible to obtain the posterior over $\mathbf{w}$ analytically, and integrate it out when making predictions.

This approximation reduces the original $\mathcal{O}\left(N^3\right)$ complexity associated with training GPs to $\mathcal{O}\left(N_{\text{RF}}^2 N\right)$, as K_{XX} is approximated by $\Phi\Phi^\top$ having rank N_{RF} (or $2N_{\text{RF}}$ for the RBF example). This matches the complexity obtained by the inducing points-based approximations when $N_{\text{RF}} = M$. On the other hand, a common complaint targeted towards this approximation is that the choice of how many random features to include must be manually tuned, going against the fully-nonparametric spirit of a standard GP. One possible counterargument is that the quality of the kernel approximation is always expected to

improve as more random features are used, indicating that N_{RF} should only be bounded by some computational budget. However, given that the mapping $\boldsymbol{\Omega}$ is jointly optimised with other model parameters with respect to the same non-convex objective function, increased parametrisation could indeed make it more difficult to converge to the global optimum.

2.3.3 Extending SSGP

Various theoretical and practical aspects of kernel approximation via random features have been explored since its proposal, including recent works by Avron et al. (2017a), Avron et al. (2017b) and Rudi and Rosasco (2017), among others. Narrowing down this vast literature to those directly related to GPs, Gal and Turner (2015) elaborate on the original SSGP model's tendency to overfit, and propose a variational inference scheme for integrating out the random features induced by $\boldsymbol{\Omega}$. The resulting model (VSSGP) is shown to yield better calibrated uncertainty estimates accompanying predictions, and a procedure for deriving the optimal weights analytically is given for the Gaussian likelihood case. Other approaches incorporating variational inference with the base SSGP model are featured in Tan et al. (2016) and Hoang et al. (2017).

In spite of the performance improvements obtained using VSSGP, given that the variational approximation is applied directly to $\boldsymbol{\Omega}$, the model posterior still deviates from the true posterior of the GP it is approximating. In the spirit of the VFE approximation introduced in Section 2.2.2, the variational Fourier features scheme proposed in Hensman et al. (2017) is intended to directly approximate the augmented GP posterior, although their initial analysis is restricted to GPs with covariance belonging to the Matérn family of kernels. Previous work on incorporating non-stationary spectral kernels in GP regression (Remes et al., 2017) was also very recently extended to include the aforementioned variational Fourier features interpretation (Shen et al., 2018). Tangentially, more efficient random feature maps have also been considered for accelerating computation, such as Fastfood approximations (Le et al., 2013) and orthogonal random features (Yu et al., 2016).

2.4 Structure Exploiting Approximations

The approximations discussed thus far do not make any assumptions regarding the structure of the underlying data. However, significant computational savings can be achieved when observations follow some structure, such as when the input space is a multidimensional grid and observations are equispaced along each individual input dimension. In

order to extend these benefits to datasets where such structure is not inherently available, research in this direction has been centred on developing projections onto structured spaces and other similar schemes. In this section, we give a brief overview of how structure exploiting approximations based on Kronecker algebra can be used for reducing the computational complexity of GPs, some aspects of which will be referenced again in subsequent chapters.

2.4.1 Grid-Structured Data

Following the exposition provided in Saatçi (2012) and Gilboa et al. (2013), we shall first consider datasets where all input points X are located on a Cartesian grid such that $X = X_1 \times \cdots \times X_{D_{\text{in}}}$, where X_d is a vector containing all distinct input locations along dimension d. Each of these may have an arbitrary number of input locations such that $X_d \in \mathbb{R}^{G_d}$, where G_d denotes the size of the input vector for dimension d. We shall also assume that the grid is complete, which entails that observations must be available for all N grid locations; it follows that $N = \prod_{d=1}^{D_{\text{in}}} G_d$. This structure arises naturally in several spatio-temporal problems such as climate modelling, where the input points generally denote latitude and longitude coordinates that can be further augmented with some periodically-spaced time dimension. Multimedia such as images and videos are also likely to inherently have such structure.

Aside from requiring a complete grid of observations, it must also be possible to decompose the designated covariance function into a product over the input dimensions. Such kernels are referred to as tensor product kernels, and may be formally specified by $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \prod_{d=1}^{D_{\text{in}}} k_d\left(x_{i,d}, x_{j,d}\right)$, where $x_{i,d}$ denotes the d^{th} dimension of $\mathbf{x}_i$, while $k_d\left(\cdot,\cdot\right)$ is a positive-definite kernel defined over a single input dimension (hence being scalar). As an illustrative example, the RBF kernel considered thus far can be interpreted as follows,

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma^2 \prod_{d=1}^{D_{\text{in}}} \exp\left(-\frac{\left(x_{i,d} - x_{j,d}\right)^2}{2l_d^2}\right), \tag{2.20}$$

thus fulfilling the aforementioned requirement. Note that not all kernels can be decomposed in this manner, in which case the algebraic speed-up detailed here cannot be directly exploited.

2.4.2 Gaussian Process Inference with Kronecker Algebra

The Kronecker product refers to a matrix multiplication operator such that if A is an $M \times N$ matrix and B has shape $P \times Q$, their Kronecker product, denoted by $A \otimes B$, is an $MP \times NQ$ matrix having the following form,

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1N}B \\ \vdots & \ddots & \vdots \\ a_{1N}B & \dots & a_{MN}B \end{bmatrix}. \tag{2.21}$$

Given a kernel decomposition as exemplified in Equation 2.20, it follows that the full covariance matrix for points in the grid can be evaluated as

$$K_{\mathrm{XX}} = K_1\left(X_1, X_1\right) \times \cdots \times K_{D_{\mathrm{in}}}\left(X_{D_{\mathrm{in}}}, X_{D_{\mathrm{in}}}\right), \tag{2.22}$$

where K_d stands for the $G_d \times G_d$ covariance matrix evaluated over the vector of scalar input locations for dimension d. As a result, K_{XX} can be rewritten using Kronecker product notation as $K_{\mathrm{XX}} = \bigotimes_{d=1}^{D_{\mathrm{in}}} K_d$.

For brevity, we directly proceed to demonstrate how GP regression may be carried out for grid inputs having spherical noise that is identical at every input location, and refer the reader to Saatçi (2012) for more detail on the properties of Kronecker algebra in this context. Recall that the most computationally expensive operations in standard GP regression involve solving linear systems with K_λ. Although it is possible to express K_{XX}^{-1} in the form of a Kronecker product, this is no longer the case when function noise is included, as is the case for K_λ. Instead, the linear system can be approximated using the following eigendecomposition,

$$K_\lambda^{-1}\mathbf{y} = \mathbf{Q}\left(\mathbf{\Lambda} + \lambda I_N\right)^{-1}\mathbf{Q}^\top\mathbf{y}, \tag{2.23}$$

where $\mathbf{Q}$ and $\mathbf{\Lambda}$ respectively denote the Kronecker products of the eigenvectors and eigenvalues of each covariance block, K_d, of the full matrix K_{XX}. This can then be solved efficiently using the following steps,

$$\alpha \leftarrow \texttt{kron\_mvprod}\left(\left[Q_1^\top, \dots, Q_{D_{\mathrm{in}}}^\top\right], \mathbf{y}\right) \tag{2.24}$$

$$\alpha \leftarrow \left(\mathbf{\Lambda} + \lambda I_N\right)^{-1}\alpha \tag{2.25}$$

$$\alpha \leftarrow \texttt{kron\_mvprod}\left(\left[Q_1, \dots, Q_{D_{\mathrm{in}}}\right], \alpha\right) \tag{2.26}$$

where `kron_mvprod` is a procedure detailed in Saatçi (2012) for efficiently computing matrix-vector multiplications involving the product of Kronecker matrices with complexity approaching $\mathcal{O}(N)$. Given that $\mathbf{\Lambda}$ is in fact a scalar vector containing the eigenvalues of the covariance blocks, the inversion operation listed in Equation 2.25 can also be evaluated with linear complexity. In consequence, full GP inference can be carried out in linear time. This methodology can be easily extended for efficiently computing the log determinant of $\boldsymbol{K}_\lambda$, as well as the associated derivatives required for optimising the GP hyperparameters. As highlighted earlier, the original formulation of this approach (Gilboa et al., 2013) relied on having observations at all possible input locations in the grid, but later extensions relaxed this condition such that missing observations and incomplete grids were also permitted (Flaxman et al., 2015; Wilson et al., 2014).

2.4.3 Structured Kernel Interpolation

The Kronecker-based methodology presented above does not involve any approximation whatsoever, and both the training and test conditionals are exact. However, most datasets will not have such structure upfront, making the application of such techniques fairly limited. In order to extend these concepts to more general problems, the notion of structured kernel interpolation (SKI) was brought to the fore in the KISS-GP approximation proposed by Wilson and Nickisch (2015). This method constrains inducing points to lie on a multidimensional grid, $\boldsymbol{U} = \boldsymbol{U}_1 \times \cdots \times \boldsymbol{U}_{D_{\text{in}}}$, where once again $\boldsymbol{U}_d$ is a vector containing all distinct inducing point locations along dimension d. Whereas computational gains from inducing point approximations typically arise from setting $M \ll N$, the computational levity of operations involving Kronecker algebra entails that a large number of inducing points can be used as long as they lie on a grid; consequently, $M \gg N$ is also feasible here.

Nevertheless, setting a large M could be problematic since approximating $\boldsymbol{K}_{\text{XX}}$ still involves multiplications with a $\boldsymbol{K}_{\text{XU}}$ matrix. So as to avoid computing $\boldsymbol{K}_{\text{XU}}$ directly, an $N \times M$ interpolating matrix, $\boldsymbol{W}_{\text{int}}$, is introduced in order to approximate this matrix as

$$\boldsymbol{K}_{\text{XU}} \approx \boldsymbol{W}_{\text{int}} \boldsymbol{K}_{\text{UU}}. \tag{2.27}$$

Given that M is expected to be large, $\boldsymbol{W}_{\text{int}}$ can be set to be very sparse, enabling fast approximation of $\boldsymbol{K}_{\text{XU}}$. Wilson and Nickisch (2015) propose to carry out cubic interpolation on the kernel evaluated on the grid of inducing points, $\boldsymbol{K}_{\text{UU}}$, such that every row of $\boldsymbol{W}_{\text{int}}$ only has four non-zero entries. Under this guise, inducing points can be reinterpreted as

being *interpolation points*; this view can also be extended to generalise the approaches described in Section 2.2, but we shall preserve this distinction in the text. Consequently, a kernel approximation using SKI can be formulated as

$$\begin{aligned} K_{\text{SKI}} &= W_{\text{int}} K_{\text{UU}} K_{\text{UU}}^{-1} K_{\text{UU}} W_{\text{int}}^{\top} \\ &= W_{\text{int}} K_{\text{UU}} W_{\text{int}}^{\top}. \end{aligned} \tag{2.28}$$

By exploiting the fast Kronecker matrix-vector multiplications introduced in Section 2.4.2, the overall complexity of this approximation becomes $\mathcal{O}\left(D_{\text{in}} M^{1+\frac{1}{D_{\text{in}}}}\right)$. Nonetheless, this approach also introduces additional design choices, such as determining the optimal density of the interpolation point grid, which require further fine-tuning than the relatively more straightforward inducing point methods. In general, the grid density is expected to be heavily dependent on the choice of kernel since more expressive kernels are likely to require a greater number of interpolation points and less sparse W_{int}.

Following its original proposition, SKI using Kronecker-based inference has become a major protagonist in the literature on GP approximation. Notable extensions include efficient algorithms for approximating the log determinant of a kernel matrix (Dong et al., 2017) and constant-time predictive distributions (Pleiss et al., 2018). SKI has also recently been used as a core design element of GPyTorch (Gardner et al., 2018), an open-source library for developing GPs in PyTorch. More laterally, SKI has also been used for deep kernel learning whereby GPs stacked upon neural network architectures can be learned end-to-end in a tractable manner (Wilson et al., 2016a,b). This particular model will feature again in our discussion on bridging the gap between GPs and deep learning in Chapter 5.

2.5 Conclusion

Developing reliable approximations for Gaussian processes is a multi-faceted challenge whereby multiple desiderata including accelerating computation, avoiding overfitting, and retaining model flexibility, must be jointly targeted while balancing all potential trade-offs. Although effective in their own right, many of the approximations discussed in this chapter introduce some degree of model approximation that could impair the overall capacity of the GP. In the next chapter, we will develop a novel methodology for introducing tractability that directly approximates the involved linear algebraic operations without introducing any changes to the model itself. By exploiting the techniques introduced in this chapter

for instead designing preconditioners suitable for kernel matrices, we put forward a flexible approach for carrying out exact GP regression and classification on a computational budget without resorting to traditional approximations.

Chapter 3

Preconditioning Kernel Matrices

As the need for large-scale kernel machines grows, much work has been directed towards scaling such models to larger datasets. At the core of most kernel machines is the need to solve linear systems involving the matrix K_λ. However, given that the dimensionality of K_λ grows with the number of data-points, N, a fundamental computational bottleneck exists: storing K_λ has $\mathcal{O}\left(N^2\right)$ complexity, while solving a linear system with K_λ is $\mathcal{O}\left(N^3\right)$. In this chapter, we describe a scalable approach to both solving kernel machines and learning their hyperparameters $\boldsymbol{\theta}$. In particular, we champion the use of preconditioned conjugate gradient solvers for Gaussian processes, and develop a broad range of preconditioners which are especially useful for evaluating linear systems involving kernel matrices. We show that this approach is exact in the limit of iterations and outperforms widely-used GP approximations for a given computational budget. The content of this chapter is predominantly based on the principal outcomes of the work presented in Cutajar et al. (2016).

3.1 Overview

The success of nonparametric models based on kernels hinges on the successful adaptation of kernel hyperparameters $\boldsymbol{\theta}$, and the motivation for preconditioning begins with an inspection of the log marginal likelihood of GP models having a prior $\mathcal{N}\left(\mathbf{f}|\mathbf{0}, K_{\mathrm{XX}}\right)$. For GPs with a Gaussian likelihood on observations, $y_i \sim \mathcal{N}\left(y_i|f_i, \lambda\right)$, we can derive analytic forms for both the marginal likelihood,

$$\log\left[p\left(\mathbf{y}|X,\boldsymbol{\theta}\right)\right] = -\frac{1}{2}\log|K_\lambda| - \frac{1}{2}\mathbf{y}^\top K_\lambda^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi, \tag{3.1}$$

and its derivatives,

$$\frac{\partial \log\left[p\left(\mathbf{y}|X,\boldsymbol{\theta}\right)\right]}{\partial \theta_i} = -\frac{1}{2}\mathrm{Tr}\left(K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}K_\lambda^{-1}\mathbf{y}, \tag{3.2}$$

where $K_\lambda = K_{\mathrm{XX}} + \lambda I_N$. The traditional approach for evaluating these expressions involves factorising the kernel matrix K_λ using the Cholesky decomposition (Benoît, 1924), which costs $\mathcal{O}\left(N^3\right)$ operations. After that, all other operations cost $\mathcal{O}\left(N^2\right)$ except for the trace term appearing in the gradient, which once again requires $\mathcal{O}\left(N^3\right)$ operations. Computations involving K_λ^{-1} also reappear in the evaluation of mean and variance predictions for test data, where a different linear system must be solved for computing the variance at every test point.

This approach is not viable for large N and, consequently, many approaches have been proposed to cheaply approximate these computations, leading to sub-optimal values for $\boldsymbol{\theta}_{\mathrm{OPT}}$ and approximate predictions. A thorough overview of such approximations was provided in the previous chapter; here we investigate the possibility of avoiding approximations altogether by arguing that for parameter optimisation it is sufficient to obtain an unbiased estimate of the gradient detailed in Equation 3.2. In particular, when such an estimate is available, it is possible to employ stochastic gradient optimisation techniques that have strong theoretical guarantees (Robbins and Monro, 1951). The problematic terms in Equation 3.2 are the solution of the linear system $K_\lambda^{-1}\mathbf{y}$ and the trace term; in this chapter, we make use of stochastic linear algebra for obtaining an unbiased approximation of the trace term as follows,

$$\mathrm{Tr}\left(K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}\right) \approx \frac{1}{N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\top} K_\lambda^{-1}\frac{\partial K_\lambda}{\partial \theta_i}\mathbf{r}^{(i)}, \tag{3.3}$$

where the $N_{\mathbf{r}}$ vectors $\mathbf{r}^{(i)}$ have components drawn from a predetermined probability distribution. This result indicates that all it takes to calculate stochastic gradients is the ability to efficiently solve linear systems. A more general overview of stochastic trace estimation and how it relates to this work is given in Section 3.2 of this chapter.

A common way to tackle the scalability of training GPs thus involves using the conjugate gradient algorithm (Hestenes and Stiefel, 1952) to solve linear systems, which relieves the constraints on both storage (the kernel matrix need not be stored) and compu-

tation (both stochastic gradients and parallelisation can be exploited). Even so, conjugate gradient is not without its issues since the conditioning of kernel matrices is often such that conjugate gradient solvers will have poor convergence in practice. A well-known approach for improving the conditioning of a matrix, which in turn accelerates convergence, is *preconditioning* (Kaasschieter, 1988). This involves the introduction of a matrix preconditioner, $\boldsymbol{P}$, which should be chosen in such a way that $\boldsymbol{P}^{-1}\boldsymbol{K}_\lambda$ approximates the identity matrix, $\boldsymbol{I}_N$. In this chapter, we apply a broad range of kernel matrix approximations as preconditioners for enabling faster GP training and inference. This allows us to exploit the important developments of approximate kernel machines to accelerate the exact computation that preconditioned conjugate gradient offers. In particular, we extend stochastic gradient learning for GPs (Anitescu et al., 2012; Filippone and Engler, 2015) by developing an unbiased estimate of the gradient for the log marginal likelihood. We make the first use of preconditioning for GP classification, and evaluate the effectiveness of our proposal over a range of problems having varying size and dimensionality. Because preconditioned conjugate gradient is exact in the limit of iterations (unlike most approximate techniques), we demonstrate a trade-off between accuracy and computational efficiency that outperforms standard approximation and factorisation approaches.

3.2 Randomised Linear Algebra

As alluded to in the prelude to this chapter, our proposal relies on recasting complex algebraic operations required for GP training and inference in terms of linear systems, which we can then solve efficiently using a principled preconditioning scheme. In Equation 3.3, we demonstrated how this can be achieved by approximating the trace term appearing in the evaluation of gradients using stochastic trace estimation. The foremost appeal of approximations based on randomised linear algebra is that we can obtain unbiased estimates for algebraic terms that would otherwise be prohibitively expensive to compute when handling large datasets. Whereas computing the trace of a general matrix $\boldsymbol{A}$ is simple (by summing all entries on the diagonal), computing the trace of some transformation of $\boldsymbol{A}$, denoted by $\rho(\boldsymbol{A})$, could be problematic if we would like to avoid computation or storage of the resulting matrix. For example, directly computing the trace term shown on the left of Equation 3.3 would require us to invert $\boldsymbol{K}_\lambda$ explicitly; conversely, using the stochastic approximation given in that same equation we can avoid this computation by solving $N_{\mathbf{r}}$ linear systems instead.

3.2.1 Stochastic Trace Estimation

Stochastic trace estimation (STE; Avron and Toledo, 2011) builds a Monte Carlo estimate of the trace of some transformed matrix $\rho(A)$ by multiplying it by a set of probing vectors $\mathbf{r}$,

$$\mathrm{Tr}(\rho(A)) \approx \frac{1}{N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\top} \rho(A)\, \mathbf{r}^{(i)}, \tag{3.4}$$

such that the expectation of $\mathbf{r}^{(i)}\mathbf{r}^{(i)\top}$ is the identity, i.e. $\mathbb{E}\left[\mathbf{r}^{(i)}\mathbf{r}^{(i)\top}\right] = I_N$. It can easily be verified that the estimated trace of $\rho(A)$ is unbiased by exploiting the cyclical property of the trace operator in the expectation of $\mathrm{Tr}\left(\mathbf{r}^{(i)\top} \rho(A)\, \mathbf{r}^{(i)}\right)$.

There are several candidate approaches for how to sample the probing vectors. The most straightforward approach involves sampling from columns of the identity matrix; however, due to poor expected sampling variance, this is not widely used in the literature. On the other hand, the Gaussian estimator involves sampling from vectors on the unit hypersphere, which significantly reduces the sample variance, but requires more random bits to generate each sample. A major progression for STE was the introduction of Hutchinson's method, which samples each element as a Bernoulli random variable requiring only a linear number of random bits, while also further reducing the sample variance. A more recent approach involves sampling from sets of mutually unbiased bases (MUBs; Fitzsimons et al., 2018), in which only a logarithmic number of bits are necessary.

In this chapter, we shall carry out STE using Hutchinson's estimator, whereby the probing vectors $\mathbf{r}$ are drawn from a Rademacher distribution where the values are either -1 or 1 with equal probability. The Bayesian log determinant approximation presented in Chapter 4 is also inspired by the concepts introduced in this section.

3.3 Iterative Methods

Iterative solver methods (Axelsson, 1994; Saad, 2003) present a practical means for solving linear systems involving large matrices without resorting to cumbersome Cholesky decompositions. Given a linear system $A\mathbf{z} = \mathbf{b}$, the solution $\mathbf{z}_\star$ is derived by sequentially updating an initial estimate $\mathbf{z}_0$ until either a budget of permitted iterations is exhausted, or more generally the norm of the residual, $\left\|A\mathbf{z}_t - \mathbf{b}\right\|$, meets some stopping criteria, where $\mathbf{z}_t$ denotes the estimated solution at iteration t. An appealing aspect of such techniques is that most solvers are proven to converge to the true solution in the limit of iterations,

and tighter convergence bounds for a finite number of iterations can often be analytically derived. On the downside however, the rate of convergence is very closely linked to the conditioning of matrix A, $\kappa(A)$, which is measured as the ratio of the largest eigenvalue to the smallest. When this is large, the slower rate of convergence may outweigh the improvements expected from such solvers vis-a-vis Cholesky decomposition.

3.3.1 Conjugate Gradient

The conjugate gradient solver (CG; Hestenes and Stiefel, 1952) is a landmark algorithm for solving linear systems. As the name implies, the search direction at every iteration of the procedure is set to be mutually orthogonal to the preceding directions, guaranteeing faster convergence than steepest gradient descent. In particular, an initial estimate $\mathbf{z}_0$, which can be set to a vector of zeros, is updated at every iteration as follows,

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \alpha_t \mathbf{s}_t, \tag{3.5}$$

where α_t denotes the length of the step to take in the search direction given by $\mathbf{s}_t$ at iteration t. The former is computed as

$$\alpha_t = \frac{\mathbf{r}_t^\top \mathbf{r}_t}{\mathbf{s}_t^\top A \mathbf{s}_t}, \tag{3.6}$$

which relies on the updated residual $\mathbf{r}_{t+1} = \mathbf{r}_t - \alpha_t A \mathbf{s}_t$. Meanwhile, the conjugate search direction $\mathbf{s}_t$ is computed as follows,

$$\mathbf{s}_{t+1} = \mathbf{r}_{t+1} + \frac{\mathbf{r}_{t+1}^\top \mathbf{r}_{t+1}}{\mathbf{r}_t^\top \mathbf{r}_t} \mathbf{s}_t. \tag{3.7}$$

The most expensive operation at each iteration is the matrix-vector multiplication appearing in the calculation of α_t and $\mathbf{r}_{t+1}$. As a result, the overall complexity of CG becomes $\mathcal{O}\left(N^2 T\right)$ where T here denotes the total number of iterations. Due to conjugacy between search directions, CG is in theory expected to converge after at most N iterations. However, in view of the difficulty in guaranteeing sufficient numerical precision, CG can potentially take much longer to converge in practice, resulting in a more costly procedure than computing the actual Cholesky decomposition of the matrix. Convergence is largely dependent on the spectrum or conditioning of the designated matrix, and an estimate of the condition number is usually a good indicator of the expected suitability of using CG.

In spite of these caveats, a key advantage of CG over Cholesky is that since the predominant matrix only ever appears in a matrix-vector multiplication, it never has to be computed directly in its entirety, which lifts an important constraint on space complexity. Another point in favour of CG is that matrix-vector products can be more easily parallelised. Pseudocode for carrying out CG is provided in Appendix A.

3.3.2 Preconditioned Conjugate Gradient

In view of the aforementioned risk of CG converging in N or more iterations, a well-established technique for accelerating convergence involves the use of *preconditioning* (PCG; Kaasschieter, 1988). This mechanism relies on the introduction of a preconditioning matrix P that transforms the original system by the preconditioned alternative

$$P^{-1}A\mathbf{z} = P^{-1}\mathbf{b}, \tag{3.8}$$

in such a way that convergence is improved over plain CG. This follows from the fact that if the conditioning of $P^{-1}A$ is lower than that of A, then $T \ll N$, giving PCG a notable speed-up compared to carrying out CG on the original linear system.

While the procedure for carrying out PCG (Algorithm 1) bears a strong resemblance to CG, a core difference is that computing the step length α and search direction $\mathbf{s}_{t+1}$ must now be adapted to incorporate the effect of P, yielding

$$\alpha_t = \frac{\mathbf{r}_t^\top P^{-1}\mathbf{r}_t}{\mathbf{s}_t^\top A\mathbf{s}_t}, \quad \text{and} \tag{3.9}$$

$$\mathbf{s}_{t+1} = P^{-1}\mathbf{r}_{t+1} + \frac{\mathbf{r}_{t+1}^\top P^{-1}\mathbf{r}_{t+1}}{\mathbf{r}_t^\top P^{-1}\mathbf{r}_t}\mathbf{s}_t. \tag{3.10}$$

The key computation in both of the above terms is the linear system $P^{-1}\mathbf{r}$. Given that P is itself an $N \times N$ matrix, this could obviously be troublesome unless P^{-1} can also be computed efficiently. To this end, devising suitable preconditioners for kernel matrices that closely approximate K_λ while remaining easy to invert is one of the primary contributions of this chapter. Otherwise, an inner CG loop will be required to solve the newly introduced linear system at every outer PCG iteration, which can turn out to be very inefficient. An illustrative comparison of the aforementioned iterative solvers is given in Figure 3.1.[1]

[1] Plots inspired by http://ikuz.eu/2015/04/15/the-concept-of-conjugate-gradient-descent-in-python.

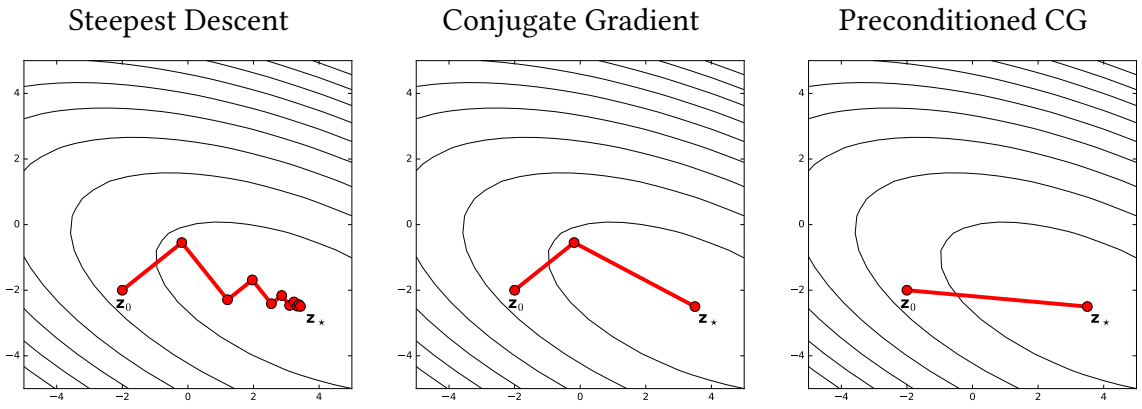


Fig. 3.1 Illustrative example showing how preconditioning enables quicker convergence to the solution $\mathbf{z}_\star$ than CG. Nonetheless, CG is still more efficient than plain steepest descent.

Algorithm 1 Preconditioned CG Algorithm, adapted from Golub and Van Loan (1996)

Require: Matrix A, preconditioner P, vector $\mathbf{b}$, convergence threshold ε, initial estimate $\mathbf{z}_0$, maximum number of iterations T

$\mathbf{r}_0 = \mathbf{b} - A\mathbf{z}_0$; $\mathbf{p}_0 = P^{-1}\mathbf{r}_0$; $\mathbf{s}_0 = \mathbf{p}_0$

for $t = 0 : T$ do

$\alpha_t = \frac{\mathbf{r}_t^\top \mathbf{p}_t}{\mathbf{s}_t^\top A \mathbf{s}_t}$

$\mathbf{z}_{t+1} = \mathbf{z}_t + \alpha_t \mathbf{s}_t$

$\mathbf{r}_{t+1} = \mathbf{r}_t - \alpha_t A \mathbf{s}_t$

if $\|\mathbf{r}_{t+1}\| < \varepsilon$ then

return $\mathbf{z}_{t+1}$

end if

$\mathbf{p}_{t+1} = P^{-1}\mathbf{r}_{t+1}$

$\beta_t = \frac{\mathbf{p}_{t+1}^\top \mathbf{r}_{t+1}}{\mathbf{p}_t^\top \mathbf{r}_t}$

$\mathbf{s}_{t+1} = \mathbf{p}_{t+1} + \beta_t \mathbf{s}_t$

end for

return $\mathbf{z}_{t+1}$

3.4 Design and Selection of Preconditioners

The advantages of using PCG over plain CG hinge on the adequacy of the preconditioning matrix $\boldsymbol{P}$ for improving the conditioning of a designated linear system, which in the case of GPs should be chosen in such a way that $\boldsymbol{P}^{-1}\boldsymbol{K}_\lambda$ approximates the identity matrix $\boldsymbol{I}_N$. Intuitively, this can be immediately obtained by setting $\boldsymbol{P} = \boldsymbol{K}_\lambda$; however, given that at each iteration of PCG we are required to solve a different linear system involving $\boldsymbol{P}$, this choice would be no easier than solving the original system. Thus, we must choose $\boldsymbol{P}$ in such a way that it approximates $\boldsymbol{K}_\lambda$ as closely as possible while also being easy to invert.

Inspired by the GP approximations detailed in Chapter 2, and extending the work by Davies (2014), here we present an assortment of kernel preconditioners for $\boldsymbol{K}_\lambda$. To avoid repetition, we refer the reader to Chapter 2 for more context on how these preconditioners arise from the literature on sparse GP approximations; in this section we directly discuss how they may be exploited for the purpose of preconditioning. Unless stated otherwise, we shall consider standard left preconditioning whereby the original problem of solving $\boldsymbol{K}_\lambda \mathbf{z} = \mathbf{v}$ is transformed by applying the preconditioner to both sides of this equation. The linear system to be solved may thus be expressed as $\boldsymbol{P}^{-1}\boldsymbol{K}_\lambda \mathbf{z} = \boldsymbol{P}^{-1}\mathbf{v}$.

3.4.1 Low-rank Preconditioners

The Nyström method was originally proposed to approximate the eigendecomposition of kernel matrices (Williams and Seeger, 2001), and offers a simple means of obtaining a low-rank approximation of $\boldsymbol{K}_{\mathrm{XX}}$. This method selects a subset of $M \ll N$ inducing points that are intended for approximating the spectrum of $\boldsymbol{K}_{\mathrm{XX}}$, and the resulting approximation is given by

$$\widetilde{\boldsymbol{K}}_{\mathrm{XX}} = \boldsymbol{K}_{\mathrm{XU}} \boldsymbol{K}_{\mathrm{UU}}^{-1} \boldsymbol{K}_{\mathrm{XU}}^{\top}, \tag{3.11}$$

where $\boldsymbol{K}_{\mathrm{UU}}$ denotes the evaluation of the kernel function over the inducing points, and $\boldsymbol{K}_{\mathrm{XU}}$ is the covariance between the full training data and the inducing points. The resulting preconditioner $\boldsymbol{P}_{\mathrm{NYS}} = \widetilde{\boldsymbol{K}}_{\mathrm{XX}} + \lambda \boldsymbol{I}_N$ can be inverted using the matrix inversion lemma, such that

$$\boldsymbol{P}_{\mathrm{NYS}}^{-1}\mathbf{v} = \frac{1}{\lambda}\left[\boldsymbol{I}_N - \boldsymbol{K}_{\mathrm{XU}}\left(\boldsymbol{K}_{\mathrm{UU}} + \boldsymbol{K}_{\mathrm{XU}}^{\top}\boldsymbol{K}_{\mathrm{XU}}\right)^{-1}\boldsymbol{K}_{\mathrm{XU}}^{\top}\right]\mathbf{v}, \tag{3.12}$$

where the inversion operation has $\mathcal{O}\left(M^3\right)$ complexity.

Fully and Partially Independent Training Conditional

The use of inducing points for approximating a GP kernel has also been utilised in the fully and partially independent training conditional approaches (FITC and PITC respectively) for scaling GP regression (Quinonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2007). In the former case, the prior covariance of the approximation leads to a preconditioner having the following form,

$$P_{\text{FITC}} = \widetilde{K}_{\text{XX}} + \text{diag}\left(K_{\text{XX}} - \widetilde{K}_{\text{XX}}\right) + \lambda I_N. \tag{3.13}$$

As its name implies, this formulation entails that the latent variables associated with the inducing points are taken to be completely conditionally independent. On the other hand, PITC entails that although inducing points assigned to a designated partition or block are conditionally independent on each other, there is no dependence between points placed in different blocks. This gives rise to the following alternative preconditioner,

$$P_{\text{PITC}} = \widetilde{K}_{\text{XX}} + \text{bldiag}\left(K_{\text{XX}} - \widetilde{K}_{\text{XX}}\right) + \lambda I_N. \tag{3.14}$$

For P_{FITC}, the diagonal resulting from the training conditional can be added to the similarly diagonal noise matrix, and the inversion lemma can be invoked as for the Nyström preconditioner. Meanwhile for P_{PITC}, the function noise λ can be added to the block diagonal matrix which can then be inverted block by block. The inversion then proceeds as before, where the inverted block diagonal matrix replaces λI_N in the original formulation.

3.4.2 Approximate Factorisation of Kernel Matrices

The following group of preconditioners gathers approximations to K_{XX} that factorise as $\widetilde{K}_{\text{XX}} = \Phi\Phi^\top$, where Φ is a low-rank $N \times M$ matrix. Here we shall consider different ways of determining Φ such that P can be inverted at a lower cost than the original kernel matrix K_λ. Once again, this enables us to employ the matrix inversion lemma, and express the solution to the linear system as

$$P^{-1}\mathbf{v} = \left(\Phi\Phi^\top + \lambda I_N\right)^{-1}\mathbf{v} = \frac{1}{\lambda}\left[I_N - \Phi\left(I_M + \Phi^\top\Phi\right)^{-1}\Phi^\top\right]\mathbf{v}. \tag{3.15}$$

We now review a selection of methods that approximate the kernel matrix K_λ in this form.

Spectral Approximation

The spectral approach uses random features for deriving a sparse approximation of a kernel (Rahimi and Recht, 2008). This was extended to GP regression by Lázaro-Gredilla et al. (2010), and relies on the assumption that stationary kernel functions can be represented as the Fourier transform of non-negative measures. As such, individual elements of K_{XX} can be approximated as follows,

$$k\left(\mathbf{x}_i, \mathbf{x}_j | \boldsymbol{\theta}\right) \approx \frac{\sigma^2}{N_{\mathrm{RF}}} \sum_{r=1}^{N_{\mathrm{RF}}} \mathbf{z}\left(\mathbf{x}_i | \widetilde{\boldsymbol{\omega}}_r\right)^{\top} \mathbf{z}\left(\mathbf{x}_j | \widetilde{\boldsymbol{\omega}}_r\right), \tag{3.16}$$

where $\mathbf{z}\left(\mathbf{x}|\widetilde{\boldsymbol{\omega}}\right) = \left[\cos\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}\right), \sin\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}\right)\right]^{\top}$ and $\widetilde{\boldsymbol{\omega}}$ denotes spectral points (or frequencies) that in the case of the RBF kernel can be sampled from $\mathcal{N}\left(\mathbf{0}, \Lambda^{-1}\right)$. As before, Λ here encodes the ARD lengthscale parameters $\left[l_1^2, \ldots, l_{D_{\mathrm{in}}}^2\right]$. The corresponding preconditioner, P_{RF}, can then be obtained by setting Φ to be an $N \times 2N_{\mathrm{RF}}$ matrix where each row takes the following form,

$$\phi\left(\mathbf{x}\right) = \left[\cos\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_1\right) \sin\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_1\right), \ldots, \cos\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_{N_{\mathrm{RF}}}\right) \sin\left(\mathbf{x}^{\top}\widetilde{\boldsymbol{\omega}}_{N_{\mathrm{RF}}}\right)\right]. \tag{3.17}$$

To the best of our knowledge, this is the first time such an approximation has been considered for the purpose of preconditioning.

Partial SVD

Another factorisation approach we consider is the partial singular value decomposition method (SVD; Golub and Van Loan, 1996). SVD factorises the original kernel matrix K_{XX} into $U\Lambda U^{\top}$, where U is a unitary matrix and Λ is a diagonal matrix of singular values. P_{SVD} can thus be constructed by setting $\Phi = U\sqrt{\Lambda}$. Given that full SVD also has $\mathcal{O}\left(N^3\right)$ complexity, we shall here consider a variation of this technique known as randomised truncated SVD (Halko et al., 2011), which constructs an approximate low-rank factorisation of K_{XX} using random sampling to accelerate computation.

Structured Kernel Interpolation

Structured kernel interpolation (SKI; Wilson et al., 2014; Wilson and Nickisch, 2015) is a kernel approximation technique that exploits the benefits of Kronecker algebra without imposing any requirements on the structure of the training data. In particular, a covariance

matrix K_{UU} is constructed over a grid of inducing points where M may even be greater than N. Instead of computing K_{XU} directly, this matrix is approximated by $W_{\mathrm{int}} K_{\mathrm{UU}}$, where W_{int} is a sparse $N \times M$ interpolation matrix for approximating K_{XU} by assigning weights to the elements of K_{UU}. In this manner, a preconditioner exploiting Kronecker algebra can be constructed as

$$\begin{aligned} P_{\mathrm{SKI}} &= W_{\mathrm{int}} K_{\mathrm{UU}} K_{\mathrm{UU}}^{-1} K_{\mathrm{UU}} W_{\mathrm{int}}^{\top} + \lambda I_N \\ &= W_{\mathrm{int}} K_{\mathrm{UU}} W_{\mathrm{int}}^{\top} + \lambda I_N. \end{aligned} \tag{3.18}$$

Recall that in the case of other preconditioners, the Woodbury inversion lemma is effective because the preconditioners are designed to have low-rank structure, entailing that the most expensive operation is the inversion of an $M \times M$ matrix, with $M \ll N$. However, this no longer holds for P_{SKI}; in this case, the matrix K_{UU} that is implicitly constructed using the Kronecker product can be quite large, possibly even larger than K_{λ}. This is necessary in order to construct a grid of inducing points that is fine-grained enough to enable effective interpolation. Nonetheless, if we apply Woodbury to P_{SKI}, we would be required to directly compute $\left(K_{\mathrm{UU}}^{-1} + W_{\mathrm{int}}^{\top} W_{\mathrm{int}}\right)^{-1}$, where the addition of $W_{\mathrm{int}}^{\top} W_{\mathrm{int}}$ no longer permits us to exploit the Kronecker structure of K_{UU} in the inversion. Instead, we are required to solve the inner linear system $P_{\mathrm{SKI}}^{-1}\mathbf{v}$ using CG, and this will have to be done for every iteration of PCG. Although the computational complexity of matrix-vector multiplications within the CG loop will be nearly linear given the structure of P_{SKI}, the number of iterations required to solve such inner linear systems can be very large for badly conditioned matrices, thus diminishing the overall benefits of preconditioning.

3.4.3 Other Approaches

Block Jacobi

An alternative to using inducing points involves constructing local GPs over segments of the original training data (Snelson and Ghahramani, 2007). An example of such an approach is the Block Jacobi approximation, which only considers a block diagonal of K_{XX} while discarding all other elements in the kernel matrix. In this manner, covariance is only expressed for points within the same block, and a preconditioner can be formed as

$$P_{\mathrm{BLD}} = \mathrm{bldiag}\left(K_{\lambda}\right) = \mathrm{bldiag}\left(K_{\mathrm{XX}}\right) + \lambda I_N. \tag{3.19}$$

Computing the inverse of this preconditioner is computationally cheap as it will also be block diagonal. However, given that a substantial amount of information contained in the original matrix is ignored, this choice is an intrinsically crude approach.

Regularisation

An appealing feature shared by the preconditioners discussed thus far (aside from SKI) is that their structure enables us to directly solve $\boldsymbol{P}^{-1}\mathbf{v}$ in an efficient manner. An alternative preconditioning technique involves adding positive regularisation, δ, to the original kernel matrix (Srinivasan et al., 2014), such that

$$\boldsymbol{P}_{\text{REG}} = \boldsymbol{K}_{\lambda} + \delta \boldsymbol{I}_{N}. \tag{3.20}$$

This follows from the fact that adding further noise to the diagonal of $\boldsymbol{K}_{\lambda}$ makes it better conditioned - the condition number is expected to decrease further as δ increases. Nonetheless, for the purpose of preconditioning, this parameter cannot be set too large since it must still be tuned in such a way that $\boldsymbol{P}_{\text{REG}}$ remains a sensible approximation to $\boldsymbol{K}_{\lambda}$.

Given that it is not possible to evaluate $\boldsymbol{P}_{\text{REG}}^{-1}\mathbf{v}$ analytically, the resulting inner linear system is solved yet again using CG, such that a linear system of equations is solved for every outer iteration of the PCG algorithm. Due to the potential loss of accuracy incurred while solving the inner linear system, a variation of standard PCG referred to as *flexible* PCG (Notay, 2000) is used instead. Using this approach, a re-orthogonalisation step is introduced at every iteration such that the search directions remain orthogonal even when the inner system is not solved to high precision. This is also done for $\boldsymbol{P}_{\text{SKI}}$.

3.5 Comparison of Preconditioners

We now provide an empirical evaluation of the preconditioners introduced in the previous section, whose properties are summarised in Table 3.1. In particular, we consider three regression datasets from the UCI repository (Asuncion and Newman, 2007), namely the Concrete dataset $\left(N = 1030, D_{\text{in}} = 8\right)$, the Powerplant dataset $\left(N = 9568, D_{\text{in}} = 4\right)$ and the Protein dataset $\left(N = 45730, D_{\text{in}} = 9\right)$. In particular, we evaluate the convergence in solving $\boldsymbol{K}_{\lambda}\mathbf{z} = \mathbf{y}$ using iterative methods, where $\mathbf{y}$ denotes the set of univariate labels of the designated dataset, and $\boldsymbol{K}_{\lambda}$ is constructed over the input data $\boldsymbol{X}$ using different configurations of kernel parameters. With this experiment, we aim to assess the quality of

different preconditioners based on how many matrix-vector products they require, which for most approaches corresponds to the number of iterations taken by PCG to converge. The convergence threshold is set to $\varepsilon^2 = N \cdot 10^{-10}$ so as to roughly accept an average error of 10^{-5} on each element of the solution.

For every variation, we configure the preconditioners so as to have a complexity lower than the $\mathcal{O}\left(N^2\right)$ cost associated with matrix-vector multiplications; by doing so, we can assume that the latter computations are the dominant cost for large N. In particular, for low-rank Nyström-based preconditioners, i.e. P_{NYS}, P_{FITC} and P_{PITC}, we set $M = \sqrt{N}$ so that when we invert the preconditioner using the matrix inversion lemma, the cost is in $\mathcal{O}\left(M^3\right) = \mathcal{O}\left(N^{\frac{3}{2}}\right)$. Similarly, for the spectral preconditioner P_{RF}, we use $M = \sqrt{N}$ random features, which will then be doubled to $2M$ for using the RBF kernel. For P_{SKI}, we take an equal number of elements on the grid for each dimension; under this assumption, Kronecker products have $\mathcal{O}\left(D_{\text{in}} N^{\frac{D_{\text{in}}+1}{D_{\text{in}}}}\right)$ complexity (Gilboa et al., 2013), and we set the size of the grid such that the complexity of applying the preconditioner also matches $\mathcal{O}\left(N^{\frac{3}{2}}\right)$. Finally, for P_{REG}, each iteration needed to apply the preconditioner in the inner CG loop requires one matrix-vector multiplication, and we add this to the outer tally of PCG iterations. For this preconditioner, we add a diagonal offset δ to the original ma-

Table 3.1 Summary of proposed preconditioners and their associated inversion strategy.

Preconditioner	**Formulation**	**Strategy**
P_{NYS}	$K_{\text{XU}} K_{\text{UU}}^{-1} K_{\text{XU}}^{\top} + \lambda I_N$	Woodbury
P_{FITC}	$K_{\text{XU}} K_{\text{UU}}^{-1} K_{\text{XU}}^{\top} + \text{diag}\left(K_{\text{XX}} - K_{\text{XU}} K_{\text{UU}}^{-1} K_{\text{XU}}^{\top}\right) + \lambda I_N$	Woodbury
P_{PITC}	$K_{\text{XU}} K_{\text{UU}}^{-1} K_{\text{XU}}^{\top} + \text{bldiag}\left(K_{\text{XX}} - K_{\text{XU}} K_{\text{UU}}^{-1} K_{\text{XU}}^{\top}\right) + \lambda I_N$	Woodbury
P_{RF}	$P_{ij} = \frac{\sigma^2}{N_{\text{RF}}} \sum_{r=1}^{N_{\text{RF}}} \cos\left[\widetilde{\boldsymbol{\omega}}_r^{\top}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right] + \lambda$	Woodbury
P_{SVD}	$K_{\text{UU}} = U \Lambda U^{\top} \Rightarrow U_{[\cdot:M]} \Lambda_{[:M,:M]} U_{[\cdot,:M]}^{\top} + \lambda I_N$	Woodbury
P_{BLD}	$\text{bldiag}\left(K_{\text{XX}}\right) + \lambda I_N$	Block Inverse
P_{SKI}	$W_{\text{int}} K_{\text{UU}} W_{\text{int}}^{\top} + \lambda I_N$ (where K_{UU} is Kronecker)	Inner CG
P_{REG}	$K_{\text{XX}} + \lambda I_N + \delta I_N$	Inner CG

trix that is equivalent to two orders of magnitude greater than λ. In general, although the asymptotic complexity of PCG is indeed no different from that of CG, we emphasise that experiencing a two-fold or five-fold (in some cases even an order of magnitude) improvement can be substantial when CG takes very long to converge or the dataset is very large.

We focus on an isotropic variant of the RBF kernel in Equation 1.6, in which a single lengthscale l is shared across dimensions, fixing the marginal variance σ^2 to one, and adding noise λ on the diagonal. For this analysis, we construct different kernels by varying the lengthscale parameter l and the noise variance λ in $\log_{10}$ scale. Results for the three selected datasets are illustrated in Figures 3.2, 3.3, and 3.4 respectively. The top part of each figure shows the number of iterations required for plain CG to converge, where we have capped the number of iterations to 100,000. Meanwhile, the bottom part of the figure represents the improvement offered by various preconditioners, measured as

$$\log_{10}\left(\frac{N_{\text{PCG}}\text{ iterations}}{N_{\text{CG}}\text{ iterations}}\right). \tag{3.21}$$

When both CG and PCG fail to converge within the upper bound, the improvement will be marked as 0, i.e. neither a gain or loss for the given budget of iterations.

Analysis

The results obtained across all datasets indicate that the low-rank preconditioners achieve notable reduction in the number of iterations required for convergence, and all approaches work best when the lengthscale is longer, characterising smoother processes. In contrast, preconditioning seems to be less effective when the lengthscale is shorter, corresponding to a kernel matrix that is more sparse. However, for instances yielding positive results, the improvement is often in the range of an order of magnitude, which is substantial when a large number of iterations is required by CG. The results also confirm that, as alluded to in the previous section, P_{BLD} is generally a poor preconditioner, particularly when the corresponding kernel matrix is dense. The only minor improvements were obtained when CG itself converges quickly, in which case preconditioning serves very little purpose either way.

Employing P_{REG} with flexible conjugate gradient does not appear to be effective in any case, particularly due to the substantial amount of iterations required for solving an inner linear system at every iteration of PCG. This implies that introducing small jitter to the diagonal does not necessarily make the system much easier to solve, while adding an

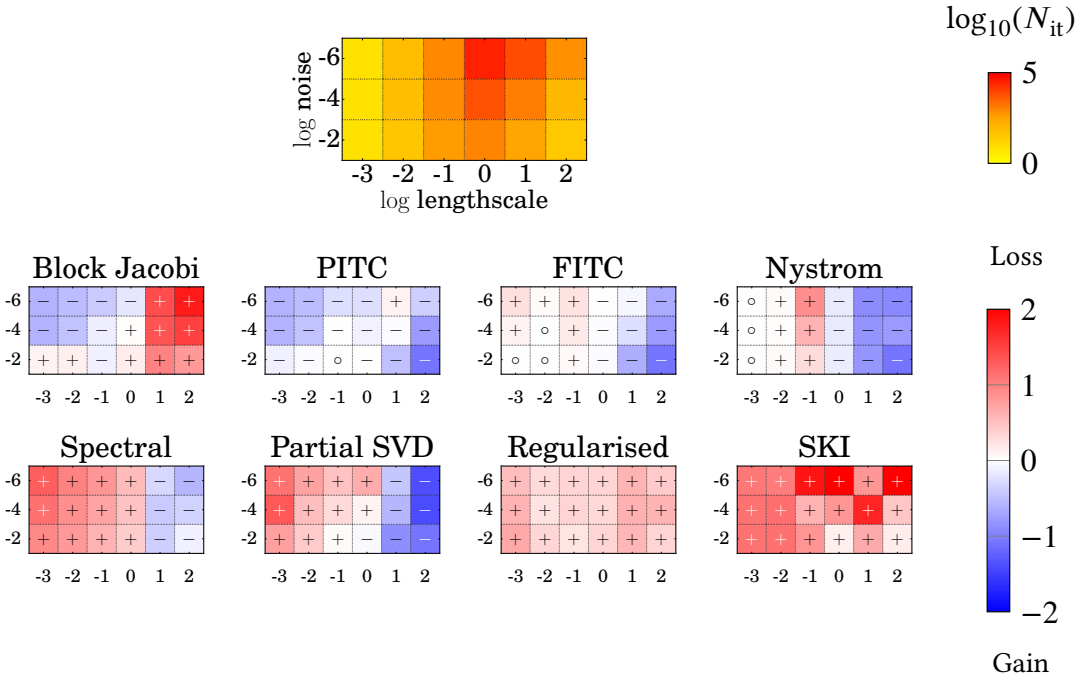


Fig. 3.2 Comparison of preconditioners for different settings of kernel parameters applied to the **Concrete** dataset. The lengthscale l and the noise variance λ are shown on the x- and y-axes respectively. The top figure indicates the number of iterations required to solve the corresponding linear system using CG, whilst the bottom part of the figure shows the rate of improvement (negative - blue) or degradation (positive - red) achieved by using PCG to solve the same linear system. Parameters and results are reported in $\log_{10}$.

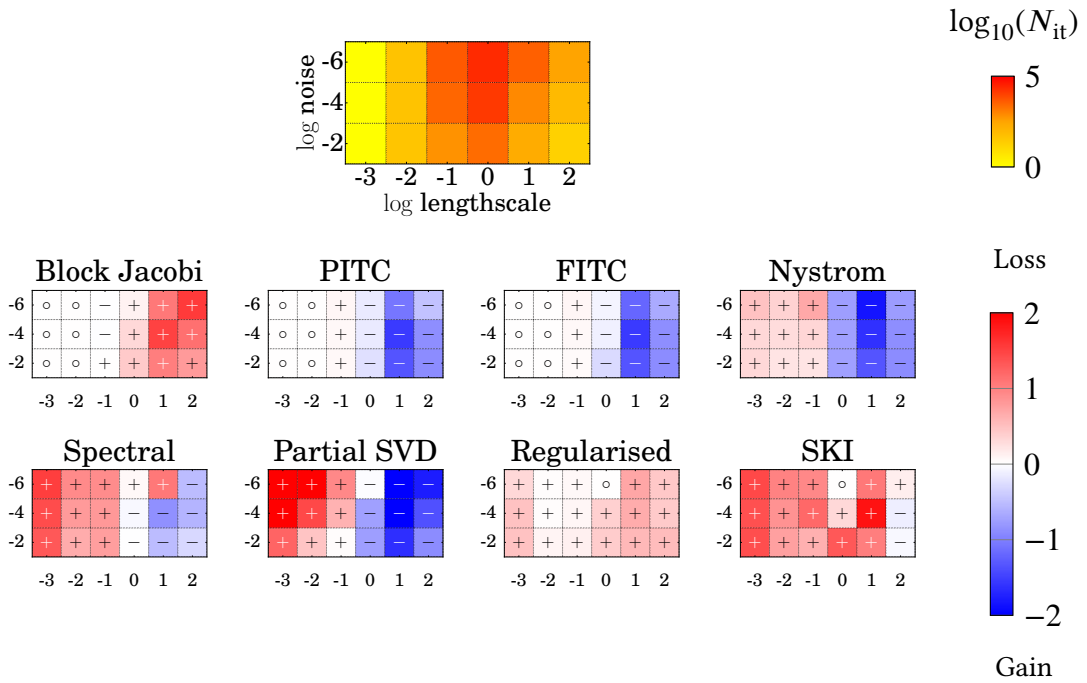


Fig. 3.3 Comparison of preconditioners for the **Powerplant** dataset.

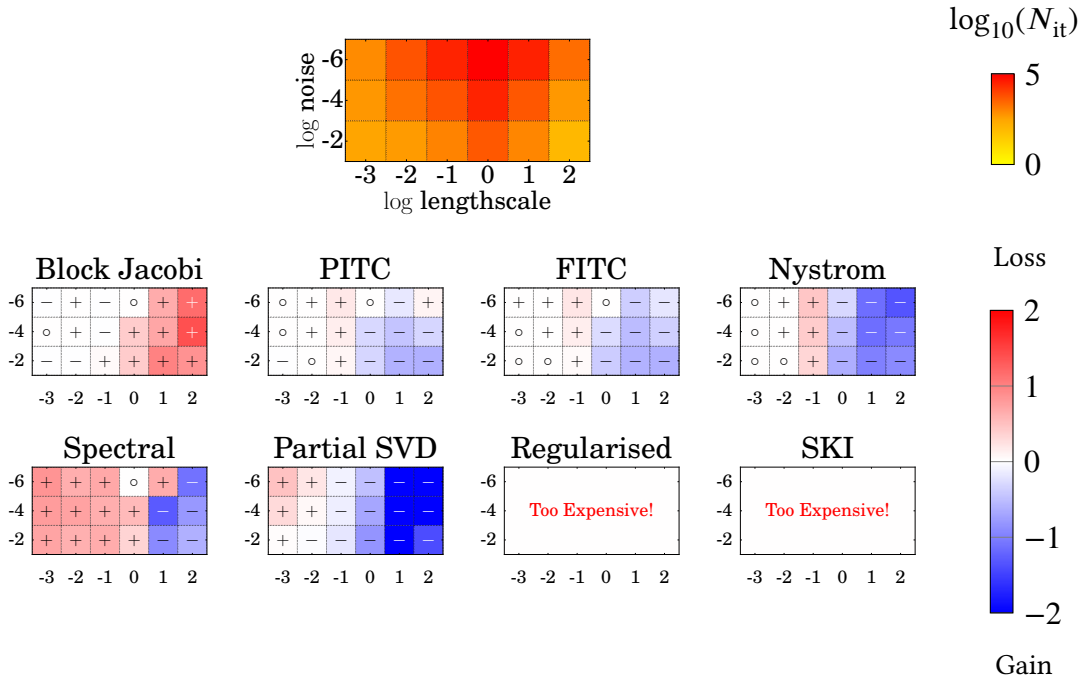


Fig. 3.4 Comparison of preconditioners for the **Protein** dataset.

overly large offset would negatively impact the convergence of the outer algorithm. Tuning this parameter could potentially result in slightly better results; however, preliminary experiments carried out in this regard yielded only minor improvement.

Finally, the results for P_{SKI} are similarly discouraging at face value. When the matrix K_λ is very badly conditioned, an excessive number of iterations for solving an inner linear system are required at every outer iteration of PCG. This greatly increases the duration of solving such systems, and as a result, this method was not included in the comparison for the Protein dataset, where it was evident from the outset that using such a preconditioner would not yield satisfactory improvement. Notwithstanding that these experiments depict a negative view of SKI preconditioning, it must be said that we assumed a fairly simplistic interpolation procedure in our experiments, where each data point was mapped to the nearest grid location. The size of the constructed grid is also hindered considerably by the constraint imposed by our upper bound on complexity. Conversely, more sophisticated interpolation strategies and grid design procedures could possibly speed up the convergence of CG for the inner systems. In line with this thought, however, one could argue that the preconditioner would no longer be straightforward to construct, which goes against our innate preference towards easily derived preconditioners.

3.6 Gaussian Processes on a Computational Budget

The primary objective of the work presented in this chapter is to reformulate the implementation of GPs in such a way that preconditioning can be effectively exploited to carry out exact training and inference on a computational budget. Although we have so far limited our discussion to regression problems, the methodology proposed here can also be extended to problems where the likelihood is non-Gaussian, thus broadening the scope of our contribution to classification tasks. In this section, we briefly demonstrate how preconditioning can be used for such problems, and evaluate the effectiveness of the proposed preconditioned GP methodology in comparison to commonly-used GP approximations for both regression and classification.

3.6.1 Preconditioning for GP Classification

In Section 1.2.4 of Chapter 1, we indicated that analytic tractability may not be preserved when the likelihood is no longer Gaussian. Among the various schemes to recover tractability in the case of models with a non-Gaussian likelihood, we choose the Laplace approxima-

tion as it can be formulated in a manner that falls back on the solution of linear systems. In particular, the Laplace approximation targets the mode of the true, albeit intractable, posterior by way of the Newton-Raphson method. For convenience, we assume that the likelihood factorises across all observations, such that $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p\left(y_i|f_i\right)$, and we consider a probit likelihood having the form

$$p\left(y_i|f_i\right) = \Phi\left(y_i f_i\right), \tag{3.22}$$

where $\Phi(\cdot)$ denotes the cumulative density function of the Gaussian distribution. As before, the latent variables $\mathbf{f}$ are assigned a GP prior with zero mean, $\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, K_{\mathrm{XX}}\right)$.

For a given setting of hyperparameters $\boldsymbol{\theta}$, we define the logarithm of the posterior density over $\mathbf{f}$ as

$$\Psi(\mathbf{f}) = \log\left[p(\mathbf{y}|\mathbf{f})\right] + \log\left[p(\mathbf{f}|\boldsymbol{\theta})\right] - \frac{N}{2}\log(2\pi). \tag{3.23}$$

Performing a Laplace approximation on this expression amounts to defining an approximate Gaussian distribution $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{f}|\hat{\mathbf{f}}, \hat{\boldsymbol{\Sigma}}\right)$ such that

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} \Psi(\mathbf{f}), \quad \text{and} \tag{3.24}$$

$$\hat{\boldsymbol{\Sigma}}^{-1} = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi\left(\hat{\mathbf{f}}\right). \tag{3.25}$$

Given that an analytic solution for the maximisation problem displayed in Equation 3.24 is not available, an iterative procedure based on the Newton-Raphson method is then employed in order to identify the posterior mode $\hat{\mathbf{f}}$. At every iteration, an initial estimate is sequentially updated as

$$\mathbf{f}^{\,\mathrm{new}} = \mathbf{f} - \left(\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f})\right)^{-1}\nabla_{\mathbf{f}}\Psi(\mathbf{f}), \tag{3.26}$$

until convergence. The gradient and the Hessian of the log of the target density can then be evaluated as

$$\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\log\left[p(\mathbf{y}|\mathbf{f})\right] - K_{\mathrm{XX}}^{-1}\mathbf{f}, \quad \text{and} \tag{3.27}$$

$$\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\log\left[p(\mathbf{y}|\mathbf{f})\right] - K_{\mathrm{XX}}^{-1} = -W - K_{\mathrm{XX}}^{-1}, \tag{3.28}$$

where we have defined $W = -\nabla_{\mathbf{f}}\nabla_{\mathbf{f}} \log\left[p\left(\mathbf{y}|\mathbf{f}\right)\right]$. Given our assumption that the likelihood factorises over observations, W is known to be a diagonal matrix. Standard algebraic manipulation then leads to

$$\mathbf{f}^{\text{new}} = \left(K_{\text{XX}}^{-1} + W\right)^{-1}\left(W\mathbf{f} + \nabla_{\mathbf{f}} \log\left[p\left(\mathbf{y}|\mathbf{f}\right)\right]\right), \tag{3.29}$$

which can be further rewritten using the matrix inversion lemma as follows,

$$\left(K_{\text{XX}}^{-1} + W\right)^{-1} = K_{\text{XX}} - K_{\text{XX}}\sqrt{W}B^{-1}\sqrt{W}K_{\text{XX}}, \quad \text{where} \tag{3.30}$$

$$B = I_N + \sqrt{W}K_{\text{XX}}\sqrt{W}. \tag{3.31}$$

Consequently, at every iteration, $\mathbf{f}^{\text{new}}$ can be reformulated as

$$\mathbf{f}^{\text{new}} = \left(K_{\text{XX}} - K_{\text{XX}}\sqrt{W}B^{-1}\sqrt{W}K_{\text{XX}}\right)\left(W\mathbf{f} + \nabla_{\mathbf{f}} \log\left[p\left(\mathbf{y}|\mathbf{f}\right)\right]\right). \tag{3.32}$$

If we additionally define $\mathbf{b} = \left(W\mathbf{f} + \nabla_{\mathbf{f}} \log\left[p\left(\mathbf{y}|\mathbf{f}\right)\right]\right)$, the above expression can be rewritten as

$$\mathbf{f}^{\text{new}} = K_{\text{XX}}\left(\mathbf{b} - \sqrt{W}B^{-1}\sqrt{W}K_{\text{XX}}\mathbf{b}\right), \tag{3.33}$$

and at convergence, we will be left with the following linear system to be solved,

$$\mathbf{a} = K_{\text{XX}}^{-1}\hat{\mathbf{f}} = \left(\mathbf{b} - \sqrt{W}B^{-1}\sqrt{W}K_{\text{XX}}\mathbf{b}\right). \tag{3.34}$$

Evaluating the featured calculations from right to left, we can observe that in order to complete a Newton-Raphson update, the most expensive operations involved are (i) the matrix-vector multiplication $K_{\text{XX}}\mathbf{b}$, (ii) solving a linear system involving B, and (iii) carrying out another matrix-vector multiplication involving K_{XX} and the vector in the parenthesis. Evaluating $\mathbf{b}$ and performing any multiplications of $\sqrt{W}$ with vectors costs $\mathcal{O}\left(N\right)$, and all these operations can be carried out without ever having to store K_{XX} or any other $N \times N$ matrix in its entirety. The use of CG for computing the Laplace approximation has been proposed elsewhere (Flaxman et al., 2015), but we make the first use of preconditioning and stochastic gradient estimation within this approach to compute stochastic gradients for non-conjugate GP models.

Stochastic Gradients for the Laplace Approximation

The Laplace approximation yields the mode $\hat{\mathbf{f}}$ of the posterior over latent variables, and offers an approximate log marginal likelihood having the following form,

$$\log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}, X\right)\right] = -\frac{1}{2}\log|B| - \frac{1}{2}\hat{\mathbf{f}}^{\top}K_{\mathrm{XX}}^{-1}\hat{\mathbf{f}} + \log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]. \tag{3.35}$$

In this exposition, we shall repeatedly make use of the following two identities in order to retain analytic tractability in the proposed methodology,

$$\log|B| = \log|I_N + \sqrt{W}K_{\mathrm{XX}}\sqrt{W}| = \log|I_N + K_{\mathrm{XX}}W|, \quad \text{and} \tag{3.36}$$

$$\left(I_N + K_{\mathrm{XX}}W\right)^{-1} = \sqrt{W}^{-1}B^{-1}\sqrt{W}. \tag{3.37}$$

The gradient of the log marginal likelihood with respect to the kernel parameters $\boldsymbol{\theta}$ requires differentiating both the terms that explicitly depend on $\boldsymbol{\theta}$, and also those that only implicitly depend on it. The latter arises because a change in the parameters also brings about a change in $\hat{\mathbf{f}}$. We thus obtain

$$\begin{aligned}\frac{\partial \log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}\right)\right]}{\partial\theta_i} = &-\frac{1}{2}\mathrm{Tr}\left(B^{-1}\frac{\partial B}{\partial\theta_i}\right)\\ &+\frac{1}{2}\hat{\mathbf{f}}^{\top}K_{\mathrm{XX}}^{-1}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}K_{\mathrm{XX}}^{-1}\hat{\mathbf{f}}\\ &+\left[\nabla_{\hat{\mathbf{f}}}\log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}\right)\right]\right]^{\top}\frac{\partial\hat{\mathbf{f}}}{\partial\theta_i}.\end{aligned} \tag{3.38}$$

Similar to the regression set-up, the trace term appearing in the derivative cannot be computed exactly for large N, so we propose a stochastic estimate as follows,

$$-\frac{1}{2}\mathrm{Tr}\left(B^{-1}\frac{\partial B}{\partial\theta_i}\right) \approx -\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}\mathbf{r}^{(i)\top}B^{-1}\frac{\partial B}{\partial\theta_i}\mathbf{r}^{(i)}. \tag{3.39}$$

By observing that the derivative of B is $\sqrt{W}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\sqrt{W}$, the above simplifies to

$$-\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}\mathbf{r}^{(i)\top}B^{-1}\sqrt{W}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\sqrt{W}\mathbf{r}^{(i)}, \tag{3.40}$$

highlighting that we are now required to solve $N_{\mathbf{r}}$ linear systems involving B instead. Meanwhile, the second term of Equation 3.38 contains the linear system $K_{\mathrm{XX}}^{-1}\hat{\mathbf{f}}$, which corresponds to the term $\mathbf{a}$ (Equation 3.34) obtained during the Laplace approximation itself.

On the other hand, the last (implicit) term appearing in the expression of the gradient can first be simplified by observing that

$$\log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}\right)\right] = \Psi\left(\hat{\mathbf{f}}\right) - \frac{1}{2}\log|B|, \tag{3.41}$$

and that the derivative of the first term with respect to $\hat{\mathbf{f}}$ is zero since this maximises $\Psi\left(\hat{\mathbf{f}}\right)$. We therefore have that

$$\left[\nabla_{\hat{\mathbf{f}}}\log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}\right)\right]\right]^{\top}\frac{\partial\hat{\mathbf{f}}}{\partial\theta_i} = -\frac{1}{2}\left[\nabla_{\hat{\mathbf{f}}}\log|B|\right]^{\top}\frac{\partial\hat{\mathbf{f}}}{\partial\theta_i}. \tag{3.42}$$

The components of $\left[\nabla_{\hat{\mathbf{f}}}\log|B|\right]$ can be obtained by considering the identity given in Equation 3.36, such that differentiating $\log|B|$ with respect to the components of $\hat{\mathbf{f}}$ becomes

$$\frac{\partial\log|I_N + K_{\mathrm{XX}}W|}{\partial\left(\hat{\mathbf{f}}\right)_j} = \mathrm{Tr}\left(\left(I_N + K_{\mathrm{XX}}W\right)^{-1}K_{\mathrm{XX}}\frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j}\right). \tag{3.43}$$

We can rewrite this by gathering K_{XX} inside the inverse itself, where this term cancels out due to the inversion of the matrix product,

$$\frac{\partial\log|I_N + K_{\mathrm{XX}}W|}{\partial\left(\hat{\mathbf{f}}\right)_j} = \mathrm{Tr}\left(\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1}\frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j}\right). \tag{3.44}$$

The resulting trace term contains the inverse of the same matrix required in the Newton-Raphson iterations of the Laplace approximation, while the matrix $\frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j}$ is zero everywhere except on the j^{th} diagonal element, where it takes the value

$$\frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j} = \frac{\partial^3\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]}{\partial\left(\hat{\mathbf{f}}\right)_j^3}. \tag{3.45}$$

In this manner, it is possible to simplify the trace term as the product between the j^{th} diagonal element of $\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1}$ and the above expression. Bearing in mind that we are required to evaluate N of these quantities, we could define

$$D = \mathrm{diag}\left[\mathrm{diag}\left[\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1}\right]\right], \quad \text{and} \tag{3.46}$$

$$(\mathbf{d})_j = \frac{\partial^3 \log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]}{\partial\left(\hat{\mathbf{f}}\right)_j^3}, \tag{3.47}$$

while rewriting

$$-\frac{1}{2}\left[\nabla_{\hat{\mathbf{f}}} \log|B|\right] = -\frac{1}{2} D\mathbf{d}. \tag{3.48}$$

This is the standard way of computing the gradient of the approximate log marginal likelihood derived using the Laplace approximation (Rasmussen and Williams, 2006). However, such an approach would be expensive to compute exactly for large N, as this would first require inverting $\left(K_{\mathrm{XX}}^{-1} + W\right)$ before computing its diagonal. Using the matrix inversion lemma would not simplify things either, since there would still be the inverse of B left to compute explicitly. We therefore aim for a stochastic estimate of this term starting from

$$\begin{aligned}\frac{\partial \log|I_N + K_{\mathrm{XX}}W|}{\partial\left(\hat{\mathbf{f}}\right)_j} &= \mathrm{Tr}\left(\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1} \frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j}\right) \\ &= \mathrm{Tr}\left(\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1} \frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j} \mathbb{E}\left[\mathbf{r}\mathbf{r}^\top\right]\right),\end{aligned} \tag{3.49}$$

where we once again introduce random probing vectors $\mathbf{r}$ with the property $\mathbb{E}\left[\mathbf{r}\mathbf{r}^\top\right] = I_N$. It follows from the cyclical property of the trace that an unbiased estimate of the trace for each component of $\hat{\mathbf{f}}$ can be computed as

$$\left[\frac{\partial \log|I_N + K_{\mathrm{XX}}W|}{\partial\left(\hat{\mathbf{f}}\right)_j}\right] \approx (\tilde{\mathbf{u}})_j = \frac{1}{N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)\top}\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1} \frac{\partial W}{\partial\left(\hat{\mathbf{f}}\right)_j} \mathbf{r}^{(i)}, \tag{3.50}$$

which entails solving $N_{\mathbf{r}}$ linear systems involving the matrix B,

$$\left(K_{\mathrm{XX}}^{-1} + W\right)^{-1} \mathbf{r}^{(i)} = K_{\mathrm{XX}}\left(\mathbf{r}^{(i)} - \sqrt{W}B^{-1}\sqrt{W}K_{\mathrm{XX}}\mathbf{r}^{(i)}\right). \tag{3.51}$$

The derivative of $\hat{\mathbf{f}}$ with respect to θ_i can then be obtained by differentiating the expression $\hat{\mathbf{f}} = K_{\mathrm{XX}}\nabla_{\hat{\mathbf{f}}} \log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]$, which is given by

$$\frac{\partial \hat{\mathbf{f}}}{\partial \theta_i} = \frac{\partial K_{\mathrm{XX}}}{\partial \theta_i}\nabla_{\hat{\mathbf{f}}} \log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right] + K_{\mathrm{XX}}\nabla_{\hat{\mathbf{f}}}\nabla_{\hat{\mathbf{f}}} \log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right] \frac{\partial \hat{\mathbf{f}}}{\partial \theta_i}. \tag{3.52}$$

Given that $\nabla_{\hat{\mathbf{f}}}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right] = -W$, we can reorganise the above equation such that

$$\left(I_N + K_{\mathrm{XX}}W\right)\frac{\partial\hat{\mathbf{f}}}{\partial\theta_i} = \frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right], \tag{3.53}$$

which yields

$$\frac{\partial\hat{\mathbf{f}}}{\partial\theta_i} = \left(I_N + K_{\mathrm{XX}}W\right)^{-1}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]. \tag{3.54}$$

Finally, the unbiased estimate of the implicit term in the gradient of the approximate log marginal likelihood becomes

$$-\frac{1}{2}\tilde{\mathbf{u}}^\top\left(I_N + K_{\mathrm{XX}}W\right)^{-1}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right]. \tag{3.55}$$

Rewriting the inverse appearing in the expression above in terms of B gives

$$-\frac{1}{2}\tilde{\mathbf{u}}^\top\sqrt{W}^{-1}B^{-1}\sqrt{W}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right], \tag{3.56}$$

and after assembling all terms back together, the stochastic approximation of the gradient presented in Equation 3.38 can be derived as

$$\begin{aligned}\frac{\partial\log\left[\hat{p}\left(\mathbf{y}|\boldsymbol{\theta}\right)\right]}{\partial\theta_i} \approx &-\frac{1}{2N_{\mathbf{r}}}\sum_{i=1}^{N_{\mathbf{r}}}\mathbf{r}^{(i)\top}B^{-1}\sqrt{W}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\sqrt{W}\mathbf{r}^{(i)}\\ &+\frac{1}{2}\mathbf{a}^\top\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\mathbf{a}\\ &-\frac{1}{2}\tilde{\mathbf{u}}^\top\sqrt{W}^{-1}B^{-1}\sqrt{W}\frac{\partial K_{\mathrm{XX}}}{\partial\theta_i}\nabla_{\hat{\mathbf{f}}}\log\left[p\left(\mathbf{y}|\hat{\mathbf{f}}\right)\right].\end{aligned} \tag{3.57}$$

Predictions

To obtain an approximate predictive distribution conditioned on a given setting of hyper-parameters $\boldsymbol{\theta}$, we can compute

$$p\left(y_\star|\mathbf{y},\boldsymbol{\theta}\right) = \int p\left(y_\star|f_\star\right)p\left(f_\star|\mathbf{f},\boldsymbol{\theta}\right)q\left(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}\right)\mathrm{d}f_\star\mathrm{d}\mathbf{f}. \tag{3.58}$$

Given the properties of multivariate normal variables, $f_\star$ is distributed as $\mathcal{N}\left(f_\star|\mu_\star,\Sigma_\star\right)$, with $\mu_\star = \mathbf{k}_\star^\top K_{\mathrm{XX}}^{-1}\mathbf{f}$ and $\Sigma_\star = k_{\star\star} - \mathbf{k}_\star^\top K_{\mathrm{XX}}^{-1}\mathbf{k}_\star$. Approximating $p\left(\mathbf{f}|\mathbf{y},\boldsymbol{\theta}\right)$ with a Gaussian

$q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta})$ makes it possible to analytically perform integration with respect to $\mathbf{f}$ in Equation 3.58. In particular, the integration yields $\mathcal{N}\left(f_\star|m_\star, s_\star^2\right)$ with

$$m_\star = \mathbf{k}_\star^\top K_{\mathrm{XX}}^{-1}\hat{\mathbf{f}}, \quad \text{and} \tag{3.59}$$

$$s_\star^2 = k_{\star\star} - \mathbf{k}_\star^\top \left(K_{\mathrm{XX}} + W^{-1}\right)^{-1}\mathbf{k}_\star. \tag{3.60}$$

These quantities can then be rewritten as $m_\star = \mathbf{k}_\star^\top \mathbf{a}$ and $s_\star^2 = k_{\star\star} - \mathbf{k}_\star^\top \sqrt{W}B^{-1}\sqrt{W}\mathbf{k}_\star$; this shows that the mean is cheap to predict, whereas the associated variance requires solving yet another linear system involving B for each test point. The univariate integration with respect to $f_\star$ follows exactly in the the case of a probit likelihood, as it is a convolution of a Gaussian and a cumulative Gaussian,

$$\int p\left(y_\star|f_\star\right)\mathcal{N}\left(f_\star|m_\star, s_\star^2\right)\mathrm{d}f_\star = \Phi\left(\frac{m_\star}{\sqrt{1+s_\star^2}}\right). \tag{3.61}$$

3.6.2 Experimental Evaluation

In order to validate the effectiveness of our proposal, we empirically report on the generalisation ability of the proposed methodology as a function of the time taken to optimise parameters $\boldsymbol{\theta}$ and compute predictions. In particular, for each of the methods featured in our comparison, we iteratively run the optimisation of kernel parameters for a set number of iterations and predict on unseen data, continually assessing how the predictive performance progresses over time for different methods. The analysis provided here is inspired by Chalupka et al. (2013), although we do not propose an approximate method to learn GP kernel parameters; instead, we put forward a means of accelerating the optimisation of $\boldsymbol{\theta}$ without any approximation[2]. Given the predictive mean and variance of $N_\star$ test points, denoted as $\mu_{\star,i}$ and $s_{\star,i}^2$ for the i^{th} test point, we report two error metrics, namely the root mean squared error (RMSE),

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_\star}\left(\mu_{\star,i} - y_{\star,i}\right)^2}, \tag{3.62}$$

along with the mean negative log likelihood (MNLL) on the test data,

[2]The one proviso to this statement is that, for non-Gaussian likelihoods, stochastic gradients target the approximate log marginal likelihood obtained by the Laplace approximation.

$$\text{MNLL} = -\frac{1}{N_\star}\sum_{i=1}^{N_\star}\log\left[p\left(y_{\star,i}|\mu_{\star,i}, s^2_{\star,i}\right)\right], \tag{3.63}$$

where $y_{\star,i}$ denotes the true label of the corresponding test point. For classification problems, we report the error rate (ERR) of the classifier instead of the RMSE,

$$\text{ERR} = 1 - \frac{1}{N_\star}\sum_{i=1}^{N_\star} r\,, \quad \text{where } r = \begin{cases} 1, & \text{if prediction is correct} \\ 0, & \text{otherwise} \end{cases}. \tag{3.64}$$

Experimental Set-up

We make use of stochastic gradients for GP models to optimise kernel hyperparameters using ADAGRAD (Duchi et al., 2011), a stochastic optimisation algorithm having a single step-size parameter. For the purpose of this experiment, we do not attempt to optimise this parameter since this would introduce additional computation. Nonetheless, our experience with training GP models using this procedure suggests that this parameter setting is not critical and we arbitrarily set the step-size to one across all experiments.

We carry out our evaluation over six datasets, three of which are regression problems, while the remaining are binary classification tasks. All results for regression and classification are shown in Figures 3.5 and 3.6 respectively. In the plots, PCG and CG refer to stochastic gradient optimisation of kernel parameters using ADAGRAD, where linear systems are solved with PCG and CG respectively. In view of the results obtained in our comparison of preconditioners, as well as its appealing ease-of-construction, we decide to proceed with the Nyström preconditioner $\boldsymbol{P}_{\text{NYS}}$. We construct the preconditioner with $\boldsymbol{M} = 4\sqrt{N}$ points randomly selected from the input data at each iteration, such that the overall complexity of the PCG method matches plain CG. For these methods, stochastic estimates of trace terms are carried out using $N_{\mathbf{r}} = 4$ random vectors. The baseline CHOL method refers to the optimisation of kernel parameters using the standard L-BFGS algorithm, where the exact log marginal likelihood and its gradients are calculated using the full Cholesky decomposition of $\boldsymbol{K}_\lambda$ (and also $\boldsymbol{B}$ for classification).

Alongside these approaches for optimising kernel hyperparameters in an exact manner, we also compare against the performance of several approximate GP methods. For this experiment, we compare against three inducing point approximations found in the software package GPstuff (Vanhatalo et al., 2013), namely the fully and partially independent training conditional approaches (FITC, PITC), and the sparse variational GP (VFE). Details

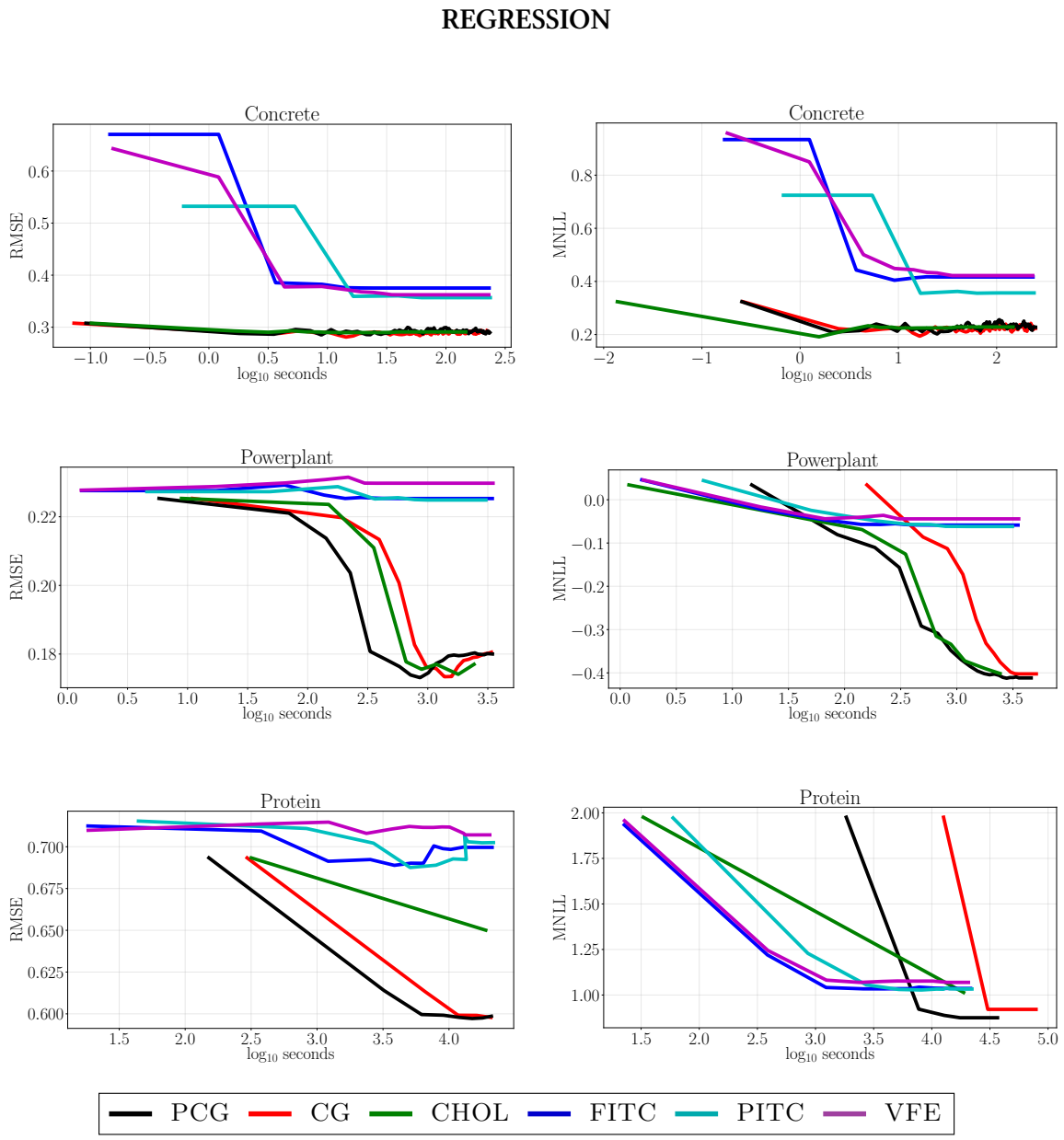


Fig. 3.5 Progression of RMSE and MNLL on held-out test data over time. The regression datasets considered have the following properties: Concrete ($N = 1030$, $D_{\text{in}} = 8$), Powerplant ($N = 9568$, $D_{\text{in}} = 4$), Protein ($N = 45730$, $D_{\text{in}} = 9$).

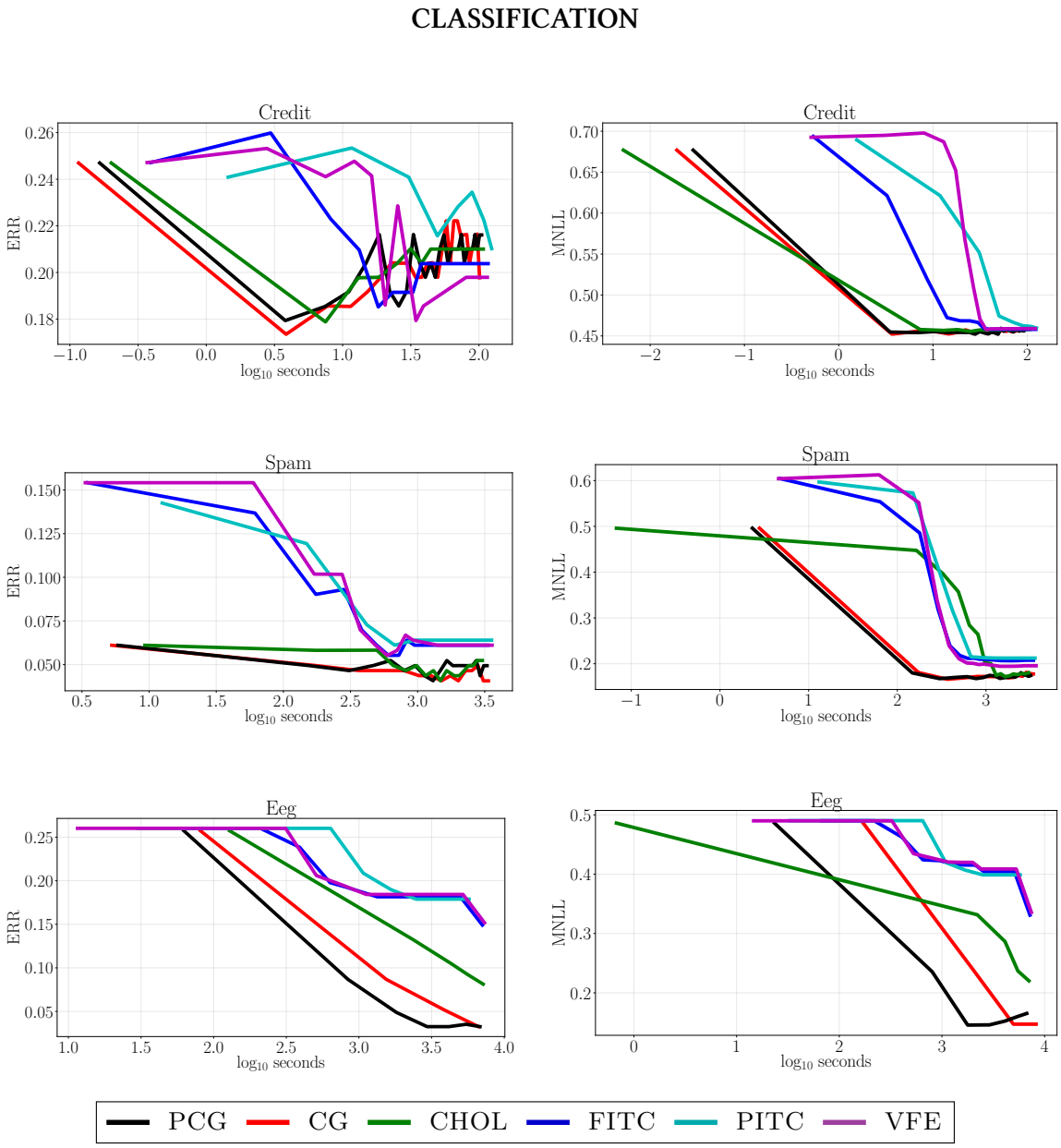


Fig. 3.6 Progression of ERR and MNLL on held-out test data over time. The classification datasets considered have the following properties: Credit ($N = 1000, D_{\text{in}} = 24$), Spam ($N = 4601, D_{\text{in}} = 57$), EEG ($N = 14979, D_{\text{in}} = 14$).

on all three approximations can be found in Chapter 2. All of the competing methods are initialised from the same set of kernel parameters, and the curves are averaged over 5 folds (this is reduced to 3 for the larger Protein and EEG datasets). For the sake of integrity, we ran each method in the comparison individually on a workstation having an Intel Xeon E5-2630 CPU with 16 cores and 128GB RAM. We also ensured that all methods reported in the comparison utilised optimised linear algebra packages and routines exploiting the multicore architecture. This diligence for ensuring fairness reinforces our assumption that the timings are not affected by external factors other than the actual implementation of the algorithms.

Analysis

For the reported experiments, it was possible to store the kernel matrix $\boldsymbol{K}_\lambda$ in memory for all datasets, making it possible to compare the competing methods against a baseline GP where computations use Cholesky decompositions. We stress that iterative approaches based on CG and PCG can be implemented without the need to store the full kernel matrix, whereas this is not possible for approaches that attempt to factorise $\boldsymbol{K}_\lambda$ exactly. It is also worth noting that for the CG and PCG approaches, calculating the log likelihood on test data entails solving one linear system for each test point; this clearly penalises the speed of these methods for the chosen experimental set-up whereby predictions are carried out every fixed number of iterations.

The results give credence to our intuition that although GP approximations quickly converge to a sensible solution, they also tend to plateau at inferior solutions to those obtained using Cholesky-based training and inference. Conversely, the performance of CG and PCG is directly comparable to the results obtained with Cholesky, while also requiring less time to converge to an optimal solution. The benefits of using PCG over CG are especially significant for the two larger datasets (Protein and EEG), where any improvement in the convergence rate of solving the involved linear systems has a markedly more pronounced effect on training time.

3.7 Conclusion

The computational and storage complexity of GPs presents the primary barrier to their scaling to large modern datasets. In this chapter we provided a comprehensive discussion of preconditioning kernel matrices, beyond existing work, and definitively illustrated that preconditioning accelerates learning in GPs without resorting to explicit approximations.

This scheme permits the use of any likelihood that factorises over observations, allowing us to tackle both regression and classification problems. We have shown robust performance improvements in both accuracy and computational cost over a selection of widely-used approximate methods for training GPs; notably, our proposal is exact in the limit of iterations. We have also demonstrated that the use of PCG is competitive with Cholesky decomposition for modestly-sized datasets where computation of Cholesky factors is still tractable. When the available data and consequently the kernel matrix grow large enough, Cholesky factorisation becomes unfeasible, positioning PCG as the best suited alternative.

Chapter 4

Towards a Probabilistic Numerics Interpretation of Gaussian Processes

Careful attention to numerical properties is essential when scaling machine learning algorithms to large datasets. In spite of the appealing uncertainty quantification inherent to Gaussian process inference, the computational or numerical uncertainty associated with these models has been largely unaccounted for in the literature. This is particularly pertinent to methodologies such as that presented in the previous chapter, where it would be desirable to capture the uncertainty tied to the approximated algebraic operations. As a step forward in the direction of incorporating probabilistic numerics with Gaussian processes, we reinterpret the problem of estimating the log determinant of large matrices as a Bayesian inference problem, yielding approximations to this term accompanied by uncertainty estimates. In conjunction with parallel work on designing probabilistic solvers for linear systems, this equips us with the necessary tools for recasting all linear algebraic operations appearing in Gaussian process inference and training as probabilistic agents. Nonetheless, we show that in spite of their theoretic appeal, state-of-the-art probabilistic numerical methods may not yet be sufficiently robust for fully-realising this concept. The first part of this chapter is adapted from joint work published in Fitzsimons et al. (2017a), while the discussion on applying probabilistic numerics to Gaussian processes is novel to this manuscript and serves as a rallying cry for future work in this direction.

4.1 Overview

Capturing model uncertainty is a core aspect of several machine learning algorithms, ranging from Bayesian neural networks (Neal, 1995) to probabilistic support vector machines (Platt, 1999). Such models are particularly suited to applications of high risk, where well-calibrated confidence measures accompanying predictions are instrumental for decision making. Aside from the uncertainty inherent to the model itself, computational limitations such as machine precision and algebraic approximations could also affect the reliability of a model's output. However, exhaustively identifying such factors can be extremely challenging, and is further hindered by the difficulty in sensibly isolating and quantifying their impact (if any) on predictive performance.

In the previous chapter, we proposed an alternative GP formulation that solves linear systems involving kernel matrices with PCG in order to accelerate training and inference. Using this framework, only the underlying algebraic operations are approximated, allowing for the GP model to remain 'exact' in the limit of unconstrained computational budgets. Nonetheless, guarantees on exactness only hold in theory (Golub and Van Loan, 1996); in practice, the quality of the approximated terms will be largely dependent on the allocated computational budget. This introduces an additional source of numerical (or computational) uncertainty within the model that is generally unaccounted for during the optimisation phase, and also when making predictions.

It follows that the overall uncertainty of GPs can be significantly underestimated if these additional sources of uncertainty are ignored. In view of the aforementioned difficulty in assessing the quality of approximated algebraic operations, *probabilistic numerics* (Hennig et al., 2015) is an emerging field of study that relies on casting linear algebraic operations as inference problems in such a way that uncertainty estimates can be returned along with the approximated solutions. Of particular interest to our set-up, probabilistic interpretations of linear solvers have recently appeared under various guises (Bartels et al., 2018; Cockayne et al., 2018; Hennig, 2015). To complement this work, and by virtue of the necessity to compute log determinants appearing in the marginal likelihood of a GP, we contribute to the existing literature on probabilistic numerics by proposing an approximation to the log determinant of large matrices that blends different elements from the literature under a Bayesian framework. In particular, we combine prior knowledge in the form of bounds from matrix theory and evidence derived from stochastic trace estimation to obtain budget-constrained probabilistic estimates for the log determinant.

The overarching goal of developing such Bayesian interpretations of linear algebra is to fuse them within more complex pipelined procedures and learning models. Coupled with existing work on probabilistic linear solvers, the proposed Bayesian log determinant approximation enables us to evaluate all algebraic operations involved in GP training and inference under a probabilistic numerics framework. The implication for GPs is that this gives us the means to obtain a distribution over the marginal likelihood and GP predictions, and we would like to carry out quadrature over samples of this distribution in order to obtain an approximation for the mean and variance of those terms. In broad terms, this amounts to introducing second-order probabilities (Ekenberg and Thorbiörnson, 2001), which can be explained as the probability of a secondary probability distribution being correct if certain information was available; in the context of probabilistic numerics, this corresponds to the probability of the secondary distribution being true if all algebraic operations were assigned an unbounded computational budget. However, although these methods work well in isolation when applied to sparse, well-conditioned matrices, this does not immediately extend to the dense, poorly-conditioned kernel matrices prevalent in the GP setting. In this light, we conclude this chapter with a cautionary note on how further work on these fundamental 'building blocks' is required before they can be effectively coupled with existing GP methodologies.

4.2 Bayesian Inference of Log Determinants

Classical linear algebra is generally viewed as a set of deterministic procedures yielding single, definitive solutions. While this holds true for most exact evaluations in the limit of unbounded computation, approximations for enabling tractability indirectly introduce a measure of approximation quality that is roughly determined by the budget allocated to the computation. With reference to conjugate gradient, the solution obtained after t iterations is expected to be less accurate than that obtained after $t + 1$ iterations, but how can one decide at run-time whether t iterations are sufficient or more should be carried out? Intuitively, it would be desirable to obtain a probability measure over the space of all possible solutions that serves as an indicator of the approximated solution's quality. Designing algorithms for effectively capturing such numerical uncertainty is the primary motivation of probabilistic numerics (Hennig et al., 2015), in which numerical computations are reinterpreted using probabilistic inference. More specifically, partial information collected during the approximation algorithm is combined with prior assumptions on the expected solution in order to derive a posterior distribution for the final objective. Tak-

ing numerical integration as an illustrative example, where the final objective is given by the integral $\int f(x)p(x)\,\mathrm{d}x$, the latent function becomes the integrand f while the partial information corresponds to evaluations of the integrand denoted by $f(x)$. In this section, we discuss a novel scheme that employs techniques inspired by probabilistic numerics to infer the log determinant of large matrices under a Bayesian framework.

4.2.1 Partial Observations by way of Stochastic Trace Estimation

In the previous chapter, we illustrated how stochastic trace estimation (STE) can be exploited for obtaining unbiased estimates of $\mathrm{Tr}\left(\rho\left(A\right)\right)$, where $\rho\left(\cdot\right)$ is some transformation applied to A. This approximation takes the form

$$\mathrm{Tr}\left(\rho\left(A\right)\right) \approx \frac{1}{N_{\mathbf{r}}} \sum_{i=1}^{N_{\mathbf{r}}} \mathbf{r}^{(i)^{\top}} \rho\left(A\right) \mathbf{r}^{(i)}, \tag{4.1}$$

where $\mathbf{r}^{(i)}$ are probing vectors sampled from a suitable estimator; in this chapter, we shall consider the Gaussian estimator whereby the $N_{\mathbf{r}}$ probing vectors are sampled from an independently and identically distributed zero-mean and unit variance Gaussian distribution. Its use in estimating the log determinant of a matrix follows from the relationship between the log determinant of A and the corresponding trace of the log-matrix,

$$\log\left|A\right| = \mathrm{Tr}\left(\log\left(A\right)\right). \tag{4.2}$$

Provided that $\log\left(A\right)$ can be properly computed, the log determinant of A can then be approximated by estimating the trace using STE. However, directly evaluating $\log\left(A\right)$ is generally infeasible, and alternative workarounds are necessary.

Given that we are primarily interested in developing a technique that is suitable for approximating log determinants appearing in the marginal likelihood of a GP, we predominantly consider covariance matrices having the form of a Gram matrix $\boldsymbol{K}$. Let us assume that $\boldsymbol{K}$ has been normalised such that the maximum eigenvalue is less than or equal to one, $\lambda_0 \leq 1$, where an upper bound on the largest eigenvalue can be estimated using Gershgorin intervals (Gershgorin, 1931). Given that kernel matrices are positive semi-definite, we also know that the smallest eigenvalue is bounded by zero, and hence $\lambda_N \geq 0$. Motivated by the identity presented in Equation 4.2, the Taylor series expansion can then be employed for evaluating the log determinant of matrices having eigenvalues bounded in this domain. In particular, this approach relies on the following identity,

$$\log\left(I_N - A\right) = -\sum_{s=1}^{\infty} \frac{A^s}{s}. \tag{4.3}$$

Naturally, the infinite summation cannot be computed in finite time, but we can approximate this term by computing a truncated series expanded up to order M instead. As the trace of matrices is additive, we thus obtain

$$\mathrm{Tr}\left(\log\left(I_N - A\right)\right) \approx -\sum_{s=1}^{M} \frac{\mathrm{Tr}\left(A^s\right)}{s}. \tag{4.4}$$

STE can then be invoked for efficiently approximating $\mathrm{Tr}\left(A^s\right)$, which in this case amounts to recursively applying M repeated matrix-vector multiplications of A by a random vector; each matrix-vector multiplication has $\mathcal{O}\left(N^2\right)$ complexity. For computing $\mathrm{Tr}\left(\log\left(K\right)\right)$, we simply set $A = I_N - K$. Such use of stochastic trace estimates in conjunction with the Taylor approximation for approximating the log determinant of a symmetric positive semi-definite matrix was explored in Boutsidis et al. (2017).

The quality of this approximation is closely related to two main factors. Firstly, in spite of returning an unbiased approximation, STE itself induces some degree of error given that only a finite number of probing vectors are used. Secondly, given that the approximation detailed above relies on an infinite summation of the Taylor series, the implied truncation may also impact the overall quality. In the absence of additive diagonal noise, the smallest eigenvalue of covariance matrices tends to be very small, which entails that A^s decays slowly as s tends towards infinity. Consequently, standard Taylor approximations to the log determinant of covariance matrices can be fairly unreliable, even when exact evaluations of $\mathrm{Tr}\left(A^s\right)$ are available. To this end, our probabilistic interpretation exploits possibly noisy approximations of $\mathrm{Tr}\left(A^s\right)$ in order to make a more informed prediction with associated uncertainty for the infinite expansion detailed in Equation 4.3.

4.2.2 The Probabilistic Numerics Approach

Probabilistic numerical algorithms are generally composed of three primary components, namely an appropriate latent function, partial or noisy observations, and a final objective. For the purpose of estimating the log determinant, the latent function corresponds to the eigenvalue distribution of A, the available data corresponds to noisy observations of $\mathrm{Tr}\left(A^s\right)$ computed using STE, while the objective is $\log|A|$ by way of the Taylor expansion given in Equation 4.3. Although the Taylor approximation to the log determinant is

perhaps not regarded as the best possible approach, we demonstrate that the intermediate trace terms obtained when raising A to higher powers may prove to be informative if these are considered to be noisy observations within a probabilistic model. Under a Bayesian framework, our proposed algorithm will thus yield both the expected approximation as well as the variance tied to the prediction.

Raw Moment Observations

Our proposal for approximating the log determinant of a matrix draws from the relation between raw moments of the eigenvalue distribution and the trace of matrices raised to higher powers, which additionally allows us to exploit STE. The raw moments of a random variable are evaluated by raising the variable to higher integer powers. Given that a function applied to a matrix is directly propagated to its eigenvalues, in the case of higher order matrices this corresponds to raising the eigenvalues to the same power. For example, the s^{th} raw moment, $\mathbf{R}^s$, of the distribution over eigenvalues for an N-dimensional matrix A can be computed as the mean eigenvalue of the matrix raised to the corresponding power,

$$\mathbf{R}^s\left[p\left(\lambda\right)\right] = \frac{1}{N}\sum_{i=1}^{N}\lambda_i^s, \tag{4.5}$$

where λ_i denotes the i^{th} eigenvalue of the matrix. The first raw moments are trivial to compute; denoting the expectation of the s^{th} raw moment as $\mathbb{E}\left[\lambda^s\right]$, we have that $\mathbb{E}\left[\lambda^0\right] = 1$, $\mathbb{E}\left[\lambda^1\right] = \frac{1}{N}\mathrm{Tr}\left(A\right)$ and $\mathbb{E}\left[\lambda^2\right] = \frac{1}{N}\sum_{i,j}A_{i,j}^2$. More generally, we can express the s^{th} raw moment as $\mathbb{E}\left[\lambda^s\right] = \frac{1}{N}\mathrm{Tr}\left(A^s\right)$, which can in turn be estimated using STE.

In view of the above, we can reformulate the log determinant approximation presented in Equation 4.2 in terms of the eigenvalues of A using the following derivation,

$$\log\left|A\right| = \sum_{i=1}^{N}\log(\lambda_i) = N\cdot\mathbb{E}\left[\log\left(\lambda\right)\right] \approx N\int\log\left(\lambda\right)p\left(\lambda\right)\mathrm{d}\lambda, \tag{4.6}$$

where the approximation is introduced due to the estimation of $p\left(\lambda\right)$. Assume that the eigenvalues are independent and identically distributed random variables drawn from $p\left(\lambda\right)$. Due to our constraints on the largest and smallest eigenvalues, this probability distribution is restricted to the domain $[0, 1]$. In this setting, we have that $\mathrm{Tr}\left(A\right) = N\cdot\mathbb{E}\left[\lambda\right]$, and more generally

$$\mathrm{Tr}\left(A^s\right) = N\cdot\mathbf{R}^s\left[p\left(\lambda\right)\right]. \tag{4.7}$$

Moreover, since the eigenvalues are assumed to lie between 0 and 1, the raw moments can be directly computed as

$$\mathbf{R}^s\left[p\left(\lambda\right)\right] = \int_0^1 \lambda^s p\left(\lambda\right) \mathrm{d}\lambda. \tag{4.8}$$

The latter formulation is particularly of interest because if a GP prior is placed over $p\left(\lambda\right)$, then the required integrals may be solved analytically using Bayesian quadrature. We will next demonstrate how noisy observations of $\mathrm{Tr}\left(A^s\right)$ corresponding to raw moments $\mathbf{R}^s\left[p\left(\lambda\right)\right]$ can be exploited in order to then make a prediction for $\sum_{s=1}^{\infty} \frac{\mathbf{R}^s[p(\lambda)]}{s}$, from which the log determinant of A can then be derived as defined in Equation 4.3.

Bayesian Quadrature

Bayesian quadrature (BQ; O'Hagan, 1991) is a well-studied procedure for performing integration of functions where analytic integration is intractable or even unavailable. As its name implies, BQ extends regular quadrature by instead returning a posterior distribution over the integral being computed. Although the scope of BQ varies between applications, in this chapter we limit our discussion to the setting where a GP prior is placed on the integrand f such that

$$Z = \int \pi(x) f(x) \mathrm{d}x, \tag{4.9}$$

where $\pi(x)$ is a known measure with respect to which the integration is carried out. Recall that in our set-up, we would like to integrate over the eigenvalue distribution of matrix A in order to estimate the expectation of $\log\left(\lambda\right)$, from which we can then obtain $\log|A|$. For enabling analytic tractability, the measure $\pi(x)$ generally takes the form of a standard Gaussian distribution; however, analytic solutions may also be derived when this takes the form of a mixture of Gaussians or polynomial. In fact, for our approximation, this measure will take the form of the polynomial λ^s associated with the raw moments of eigenvalues.

A full discussion of BQ may be found in O'Hagan (1991) and Rasmussen and Ghahramani (2002); for the scope of this thesis, we simply restate the following expressions, adapted from Huszar and Duvenaud (2012), for computing mean and variance predictions of Z under a BQ framework,

$$\begin{aligned}\mathbb{E}[Z] &= \mathbb{E}\left[\int \pi(x) f(x) \mathrm{d}x\right] \\ &= \iint \pi(x) f(x) p\left(f(x) | f(X)\right) \mathrm{d}x \mathrm{d}f \\ &= \int \pi(x) \overline{f}(x) \mathrm{d}x \\ &= \left[\int \pi(x) k\left(x, X\right) \mathrm{d}x\right]^{\top} \boldsymbol{K}(X, X)^{-1} f\left(X\right), \end{aligned} \tag{4.10}$$

$$\begin{aligned}\mathbb{V}[Z] = &\left[\iint \pi(x) k\left(x, x\right) \pi(x) \mathrm{d}x \mathrm{d}x\right] - \\ &\left[\int \pi(x) k\left(x, X\right) \mathrm{d}x\right]^{\top} K\left(X, X\right)^{-1} \left[\int \pi(x) k\left(x, X\right) \mathrm{d}x\right], \end{aligned} \tag{4.11}$$

where $f(X)$ denotes available observations at input locations X, and $\overline{f}(x)$ denotes the posterior of f at x. The key terms in the above equations are the expressions for computing covariances involving integrals,

$$k\left(\int \pi(x) \cdot \, \mathrm{d}x, x'\right) = \int \pi(x) k\left(x, x'\right) \mathrm{d}x, \quad \text{and} \tag{4.12}$$

$$k\left(\int \pi(x) \cdot \, \mathrm{d}x, \int \pi(x') \cdot \, \mathrm{d}x'\right) = \iint \pi(x) k\left(x, x'\right) \pi(x') \mathrm{d}x \mathrm{d}x', \tag{4.13}$$

which, as we discuss next, are generally intractable unless the choice of kernel yields analytic solutions.

4.2.3 Inference on the Log Determinant

In our proposal, we place a GP prior on the distribution of eigenvalues $p(\lambda)$, and condition this GP on observations having the form $\langle \mathbf{R}^s\left[p(\lambda)\right], \frac{1}{N}\mathrm{Tr}\left(A^s\right)\rangle$, where $\mathrm{Tr}\left(A^s\right)$ is approximated using STE for $s > 2$. For notational convenience, we henceforth denote $\mathbf{R}^s\left[p(\lambda)\right]$ as $\mathbf{R}^s_\lambda$. Following the definition of raw moments given in Equation 4.8, computing the covariance between raw moments requires us to integrate the kernel with respect to the corresponding polynomials in λ,

$$k\left(\mathbf{R}_{\lambda}^{s}, \mathbf{R}_{\lambda'}^{s'}\right) = \int_0^1 \int_0^1 \lambda^s k\left(\lambda, \lambda'\right) \lambda'^{s'} \mathrm{d}\lambda \mathrm{d}\lambda'. \tag{4.14}$$

Given that we intend to use the GP to make a prediction for $\log(\lambda)$, we are also required to compute integrals of the form

$$k\left(\log(\lambda), \mathbf{R}_{\lambda'}^{s'}\right) = \int_0^1 \int_0^1 \log(\lambda)\, k\left(\lambda, \lambda'\right) \lambda'^{s'} \mathrm{d}\lambda \mathrm{d}\lambda', \quad \text{and} \tag{4.15}$$

$$k\left(\log(\lambda), \log\left(\lambda'\right)\right) = \int_0^1 \int_0^1 \log(\lambda)\, k\left(\lambda, \lambda'\right) \log\left(\lambda'\right) \mathrm{d}\lambda \mathrm{d}\lambda'. \tag{4.16}$$

Using the identity presented in Equation 4.3, these two covariances can be more practically formulated in terms of raw moments,

$$k\left(\sum_{s=1}^{\infty} \frac{\mathbf{R}_{\lambda}^{s}}{s}, \mathbf{R}_{\lambda'}^{s'}\right) \qquad \text{and} \qquad k\left(\sum_{s=1}^{\infty} \frac{\mathbf{R}_{\lambda}^{s}}{s}, \sum_{s'=1}^{\infty} \frac{\mathbf{R}_{\lambda'}^{s'}}{s'}\right).$$

The covariance expressed in Equation 4.14 is required to compute entries of the $M \times M$ matrix evaluated on the M exact or stochastic (for $s > 2$) observations of $\mathrm{Tr}\left(A^s\right)$. On the other hand, Equations 4.15 and 4.16 are necessary for evaluating the $\mathbf{k}_{\star}$ and $k_{\star\star}$ terms appearing when making predictions, as was previously shown in Equations 4.10 and 4.11.

In general, the integrals introduced above cannot be computed analytically, and it is necessary to develop custom kernels having an analytic form allowing for efficient computation. To this purpose, in this section we describe a histogram kernel for computing the indicated covariances between raw moments and ultimately $\log(\lambda)$ itself. Subsequently, we demonstrate how the inclusion of a prior mean on the GP used to model $p(\lambda)$, as well as truncation by way of theoretic bounds on the log determinant, can be exploited for improving our approximation. We conclude this section by summarising the steps involved in estimating log determinants using this methodology, and comment about its computational complexity.

Histogram Kernel

To enable analytic tractability, we develop a histogram kernel resembling a piecewise constant kernel, that is computed as

$$k\left(x, x'\right) = \sum_{j=0}^{m-1} \mathcal{H}\left(\frac{j}{m}, \frac{j+1}{m}, x, x'\right), \tag{4.17}$$

where

$$\mathcal{H}\left(\frac{j}{m}, \frac{j+1}{m}, x, x'\right) = \begin{cases} 1 & x, x' \in \left[\frac{j}{m}, \frac{j+1}{m}\right] \\ 0 & \text{otherwise} \end{cases}. \tag{4.18}$$

In the above, m denotes the number of bins in the range $[0, 1]$ (implying bin size of $\frac{1}{m}$), while j indexes these bins. The covariance between a raw moment and definite eigenvalue can then be evaluated as

$$\begin{aligned} k\left(\mathbf{R}^s_\lambda, \lambda'\right) &= \int_0^1 \lambda^s k\left(\lambda, \lambda'\right) \mathrm{d}\lambda \\ &= \frac{1}{s+1}\left(\left(\frac{j+1}{m}\right)^{s+1} - \left(\frac{j}{m}\right)^{s+1}\right), \end{aligned} \tag{4.19}$$

when λ' lies in the interval $\left[\frac{j}{m}, \frac{j+1}{m}\right]$. Extending this to the covariance function between raw moments yields

$$\begin{aligned} k\left(\mathbf{R}^s_\lambda, \mathbf{R}^{s'}_{\lambda'}\right) &= \int_0^1 \int_0^1 \lambda^s k\left(\lambda, \lambda'\right) {\lambda'}^{s'} \mathrm{d}\lambda \mathrm{d}\lambda' \\ &= \sum_{j=0}^{m-1} \prod_{\bar{s} \in (s, s')} \frac{1}{\bar{s}+1}\left(\left(\frac{j+1}{m}\right)^{\bar{s}+1} - \left(\frac{j}{m}\right)^{\bar{s}+1}\right). \end{aligned} \tag{4.20}$$

This formulation of a kernel between noisy observations of raw moments compactly allows us to perform inference over $p(\lambda)$. However, the ultimate goal is to predict $\log|A|$ by way of $\sum_{s=1}^{\infty} \frac{\mathrm{Tr}(A^s)}{s}$. Although this seemingly requires a more complex set of kernel expressions, by extracting the implied infinite summations from the kernel function, we can also obtain the following closed form solutions for these terms,

$$
\begin{aligned}
k\left(\sum_{s=1}^{\infty}\frac{\mathbf{R}_{\lambda}^{s}}{s},\mathbf{R}_{\lambda'}^{s'}\right) &= \sum_{s=1}^{\infty}\frac{1}{s}k\left(\mathbf{R}_{\lambda}^{s},\mathbf{R}_{\lambda'}^{s'}\right)\\
&= \sum_{j=0}^{m-1}\frac{1}{s'+1}\left(\left(\frac{j+1}{m}\right)^{s'+1}-\left(\frac{j}{m}\right)^{s'+1}\right)\times\\
&\sum_{s=1}^{\infty}\left(\frac{1}{s\left(s+1\right)}\left(\frac{j+1}{m}\right)^{s+1}-\left(\frac{j}{m}\right)^{s+1}\right)\\
&= \sum_{j=0}^{m-1}\frac{1}{s'+1}\left(\left(\frac{j+1}{m}\right)^{s'+1}-\left(\frac{j}{m}\right)^{s'+1}\right)\left(S\left(\frac{j+1}{m}\right)-S\left(\frac{j}{m}\right)\right),
\end{aligned}
\tag{4.21}
$$

and

$$
\begin{aligned}
k\left(\sum_{s=1}^{\infty}\frac{\mathbf{R}_{\lambda}^{s}}{s},\sum_{s'=1}^{\infty}\frac{\mathbf{R}_{\lambda'}^{s'}}{s'}\right) &= \sum_{s'=1}^{\infty}\frac{1}{s'}k\left(\sum_{s=1}^{\infty}\frac{\mathbf{R}_{\lambda}^{s}}{s},\mathbf{R}_{\lambda'}^{s'}\right)\\
&= \sum_{j=0}^{m-1}\left(S\left(\frac{j+1}{m}\right)-S\left(\frac{j}{m}\right)\right)^{2},
\end{aligned}
\tag{4.22}
$$

where $S\left(\alpha\right)=\sum_{s=1}^{\infty}\frac{\alpha^{s+1}}{s(s+1)}$. When $0<\alpha<1$, this can conveniently be evaluated exactly as

$$
S\left(\alpha\right)=\alpha+\left(1-\alpha\right)\log\left(1-\alpha\right). \tag{4.23}
$$

The derivations presented above enable us to compute the log determinant approximation, and its associated variance, by replacing Equations 4.21 and 4.22 in the formulations given in Equations 4.10 and 4.11 respectively. The entries of $K(X,X)$ are evaluated using the covariance between raw moments given in Equation 4.20.

Incorporating a Prior Mean Function

While GPs, and in this case BQ, can be formulated with a zero-mean prior without loss of generality, setting a mean function is useful for incorporating additional prior information

on the distribution being modelled. If $p(\lambda)$ is constructed as the summation of a constant mean function, $g(\lambda)$, and a GP, $f(\lambda)$, for modelling the residual, we obtain

$$p(\lambda) = g(\lambda) + f(\lambda), \tag{4.24}$$

and the previously derived moment observations may be decomposed into

$$\int \lambda^s p(\lambda)\,\mathrm{d}\lambda = \int \lambda^s g(\lambda)\,\mathrm{d}\lambda + \int \lambda^s f(\lambda)\,\mathrm{d}\lambda. \tag{4.25}$$

Given that λ is constrained to lie between 0 and 1, here it makes sense to set a beta distribution as the prior mean. This has two convenient properties: firstly, it is fully specified by the mean and variance of the distribution, which can be computed using the trace and Frobenius norm of the matrix; secondly, the s^{th} raw moment of a Beta distribution parametrised by α and β is given by

$$\mathbf{R}^s[g(\lambda)] = \frac{\alpha + s}{\alpha + \beta + s}, \tag{4.26}$$

which is straightforward to compute. Furthermore, using known identities, we can immediately derive the prior on $\log(\lambda)$ itself as

$$\mathbb{E}\left[\log(g(\lambda))\right] = \psi(\alpha) - \psi(\alpha + \beta), \tag{4.27}$$

where $\psi(\cdot)$ is the digamma function. This can then be added to the posterior expectation of the log determinant which is computed as in Equation 4.10.

The GP prior being proposed for modelling the density of eigenvalues does not guarantee a strictly positive function, and consequently there may be prior mass placed on negative functions that do not constitute valid densities. To this end, including a prior beta distribution has the additional benefit of shifting the probability mass of the GP towards the positive domain. Coupling the aforementioned prior with the raw moment observations used for training the model generally circumvents issues arising from this misspecification in practice.

Incorporating Bounds on the Log Determinant

Borrowing from the literature on bounds for the log determinant of matrices (Golub and Van Loan, 1996), we can exploit such upper and lower bounds to truncate the resulting predictive distribution to the relevant domain, which should additionally provide greater stability in the occurrence of predictions which are very incorrect. Given upper and lower

bounds respectively denoted by U_{B} and L_{B}, the posterior predictive distribution with mean and variance $\widehat{\mu}$ and $\widehat{\sigma^2}$ can be truncated as

$$\widehat{\mu}_{\mathrm{T}} = \widehat{\mu} + \frac{\phi(a) - \phi(b)}{\Phi(b) - \Phi(a)}\widehat{\sigma}, \quad \text{and} \tag{4.28}$$

$$\widehat{\sigma^2}_{\mathrm{T}} = \widehat{\sigma^2}\left[1 + \frac{a\phi(a) - b\phi(b)}{\Phi(b) - \Phi(a)} - \left(\frac{\phi(a) - \phi(b)}{\Phi(b) - \Phi(a)}\right)^2\right], \tag{4.29}$$

where $a = \frac{L_{\mathrm{B}} - \widehat{\mu}}{\widehat{\sigma}}$, $b = \frac{U_{\mathrm{B}} - \widehat{\mu}}{\widehat{\sigma}}$, while $\phi(\cdot)$ and $\Phi(\cdot)$ respectively denote the probability density and the cumulative distribution functions of the Gaussian distribution.

These additional constraints can also be propagated to the hyperparameter optimisation procedure by incorporating them into the likelihood function via the product rule, resulting in the following truncated marginal likelihood,

$$\mathcal{L}_{\mathrm{TML}} = \mathcal{L}_{\mathrm{ML}} + \log\left(\Phi\left(\frac{U_{\mathrm{B}} - \widehat{\mu}}{\widehat{\sigma}}\right) - \Phi\left(\frac{L_{\mathrm{B}} - \widehat{\mu}}{\widehat{\sigma}}\right)\right). \tag{4.30}$$

Let A be an $N \times N$ symmetric positive definite matrix, where $\mu_1 = \mathrm{Tr}(A)$, $\mu_2 = \|A\|_F^2$ and $\lambda_i(A) \in \left[\lambda_{\mathrm{L}}, \lambda_{\mathrm{U}}\right]$ with $\lambda_{\mathrm{L}} > 0$. The bounds on the trace of the log determinant for such a matrix are given by

$$\begin{bmatrix} \log\left(\lambda_{\mathrm{L}}\right) \\ \log(t) \end{bmatrix}^{\top} \begin{bmatrix} \lambda_{\mathrm{L}} & t \\ \lambda_{\mathrm{L}}^2 & t^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \leq \mathrm{Tr}(\log|A|) \leq \begin{bmatrix} \log\left(\lambda_{\mathrm{U}}\right) \\ \log(\bar{t}) \end{bmatrix}^{\top} \begin{bmatrix} \lambda_{\mathrm{U}} & \bar{t} \\ \lambda_{\mathrm{U}}^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{4.31}$$

where,

$$t = \frac{\lambda_{\mathrm{L}} \cdot \mu_1 - \mu_2}{\lambda_{\mathrm{L}} \cdot N - \mu_1}, \quad \text{and} \tag{4.32}$$

$$\bar{t} = \frac{\lambda_{\mathrm{U}} \cdot \mu_1 - \mu_2}{\lambda_{\mathrm{U}} \cdot N - \mu_1}. \tag{4.33}$$

This procedure necessitates first estimating bounds on the maximum and minimum eigenvalues of the matrix, and we do so using Gershgorin intervals (Gershgorin, 1931). The bounds on the log determinant itself can then be easily computed given that both the trace and Frobenius norm can be computed exactly.

Algorithm 2 A probabilistic numerics approach for estimating log determinants

Input: Positive semi-definite matrix A, histogram kernel $k(\cdot,\cdot)$ with initial parameters $\boldsymbol{\theta}$, expansion order M, and random probing vectors $\mathbf{r}$

Output: Truncated posterior mean μ_{T}, and uncertainty σ^2_{T}

1: $A \leftarrow \textsc{normalise}(A)$
2: $L_{\mathrm{B}}, U_{\mathrm{B}} \leftarrow \textsc{getBounds}(A)$
3: for $i \leftarrow 1$ to M do
4: $\quad \mathbf{y}_i \leftarrow \mathrm{STE}(A, i, \mathbf{r})$
5: end for
6: for $i \leftarrow 1$ to M do
7: $\quad$ for $j \leftarrow 1$ to M do
8: $\quad\quad K_{ij} \leftarrow k\left(\mathrm{R}^i_\lambda, \mathrm{R}^j_\lambda; \boldsymbol{\theta}\right)$
9: $\quad$ end for
10: end for
11: $\boldsymbol{\theta}_{\mathrm{OPT}}, K_{\mathrm{XX}} \leftarrow \textsc{tuneKernel}\left(K_{\mathrm{XX}}, \mathbf{y}, \left[L_{\mathrm{B}}, U_{\mathrm{B}}\right]\right)$
12: for $i \leftarrow 1$ to M do
13: $\quad \mathbf{k}_{\star i} \leftarrow k\left(\log(\lambda), \mathrm{R}^i_\lambda; \boldsymbol{\theta}_{\mathrm{OPT}}\right)$
14: end for
15: $k_{\star\star} \leftarrow k\left(\log(\lambda), \log(\lambda); \boldsymbol{\theta}_{\mathrm{OPT}}\right)$
16: $\widehat{\mu}, \widehat{\sigma^2} \leftarrow \textsc{gpPred}\left(\mathbf{y}, K_{\mathrm{XX}}, \mathbf{k}_\star, k_{\star\star}\right)$
17: $\widehat{\mu}_{\mathrm{T}}, \widehat{\sigma^2}_{\mathrm{T}} \leftarrow \textsc{truncate}\left(\widehat{\mu}, \widehat{\sigma^2}, \left[L_{\mathrm{B}}, U_{\mathrm{B}}\right]\right)$

4.2.4 Algorithm Complexity and Recap

Due to its cubic complexity, GP inference is typically considered detrimental to the scalability of a model. However, in our formulation, the GP is only built upon the noisy observations of $\mathrm{Tr}\left(A^s\right)$, which rarely exceed few tens of points. As a result, given that we can assume this to be orders of magnitude smaller than the dimensionality N of the matrix A, the computational complexity is dominated by the matrix-vector multiplications involved in STE, i.e. $\mathcal{O}\left(N^2\right)$ for dense matrices.

The steps involved in the procedure described within this section are unified and summarised as pseudocode in Algorithm 2. The input matrix A is first normalised by using Gershgorin intervals to find the largest eigenvalue (line 1), while the expected bounds on the log determinant (line 2) are calculated using the derivation detailed in Equation 4.31. The noisy approximations of $\mathrm{Tr}\left(A^s\right)$ up to expansion order M (lines 3-5), denoted here

as $\mathbf{y}$, are then obtained by way of STE as described in Section 4.2.1. We can then place a GP prior on these observations, where the entries of the kernel matrix are computed using Equation 4.20 (lines 6-10). Subsequently, the kernel parameters are tuned by optimising the model with respect to the marginal likelihood (line 11), as indicated in the previous subsection.

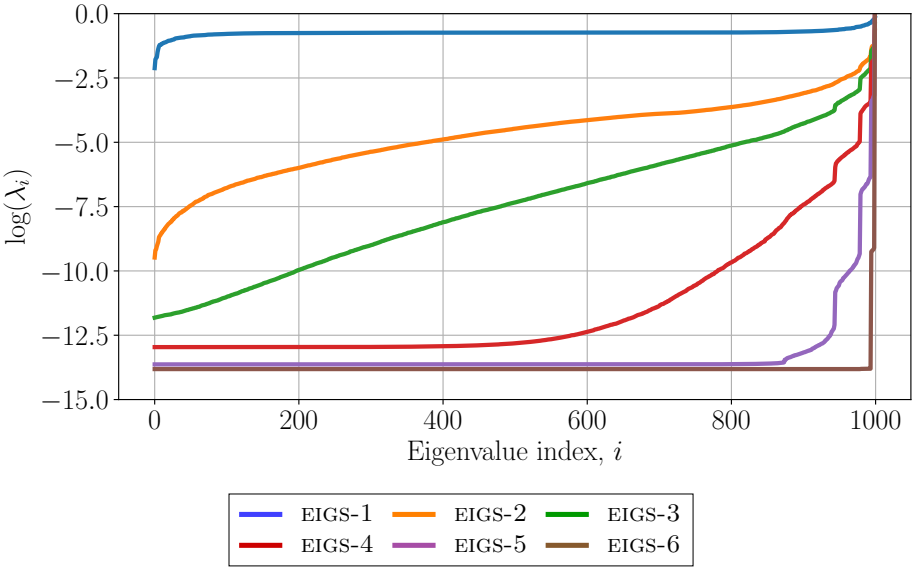
Recall that we ultimately seek to make a prediction for the infinite Taylor approximation, and hence the exact log determinant. To this end, we must compute the covariance between $\log(\lambda)$ and the observed raw moments, given by $\mathbf{k}_\star$ (lines 12-14), as well as the self-covariance $k_{\star\star}$ (line 15) using Equations 4.21 and 4.22 respectively. The posterior mean and variance (line 16) may then be evaluated by completing Equations 4.10 and 4.11. As outlined in Equations 4.28 and 4.29, the resulting posterior distribution can then be truncated using the derived bounds in order to obtain the final approximation of the log determinant and its uncertainty (line 17).

4.3 Evaluation

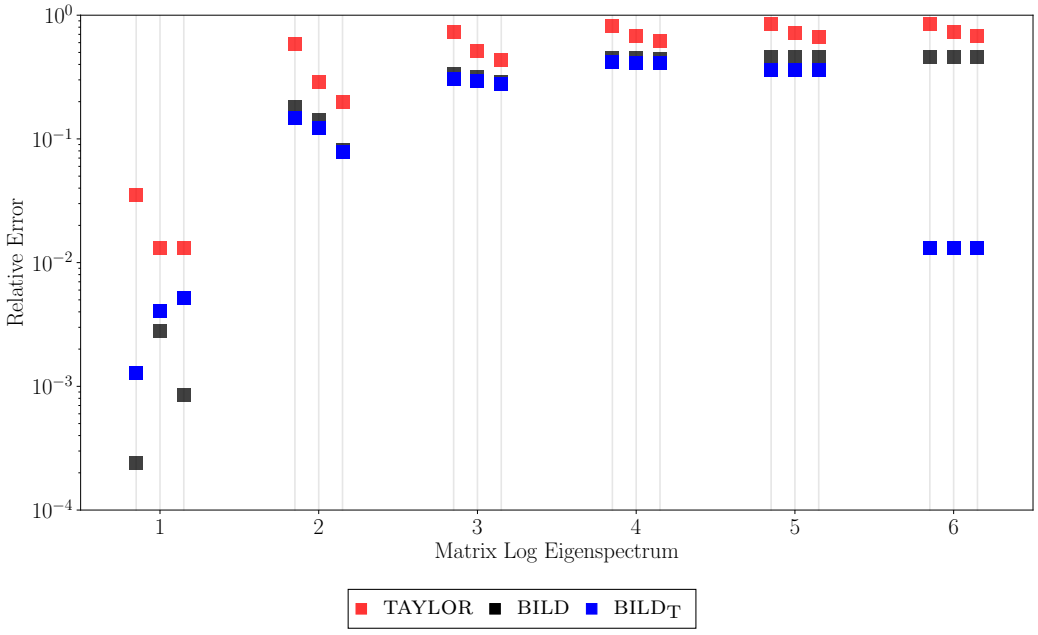
In this section, we proceed to validate the effectiveness of our proposal and demonstrate how the appeal of this formulation extends beyond its intrinsic novelty, whereby its performance is shown to be directly comparable to the standard Taylor approximation upon which this method is based. We set up a variety of experiments for assessing the proposed model's performance, including both synthetically-constructed and real matrices. In view of the model's probabilistic formulation, we also include an additional experiment for assessing the quality of uncertainty estimates returned by the model. We shall henceforth refer to the standard approximation presented here as BILD, and the truncated variation as BILD_T. We conclude this section by briefly mentioning an alternative novel approximation to the log determinant of large matrices that showcases similarly good performance, albeit without the desirable uncertainty measures returned by BILD.

4.3.1 Synthetically-constructed Matrices

In the previous chapter, we observed how the suitability of preconditioning for solving linear systems is closely tied to the conditioning of the involved matrix. Similarly, the quality of the log determinant approximation proposed here is intrinsically linked to the decay rate of the matrix's eigenvalues; this was also reported in other work on approximating log determinants (Ubaru et al., 2017). We investigate this characteristic by setting



(a) Log eigenspectrum of the six synthetically-constructed matrices.



(b) Relative error achieved by competing approximations given computational budgets of 5, 25, and 50 $\mathcal{O}\left(N^2\right)$ matrix-vector multiplications (left to right respectively).

Fig. 4.1 Comparison of log determinant approximations over six synthetically-constructed matrices with different eigenvalue decay rates.

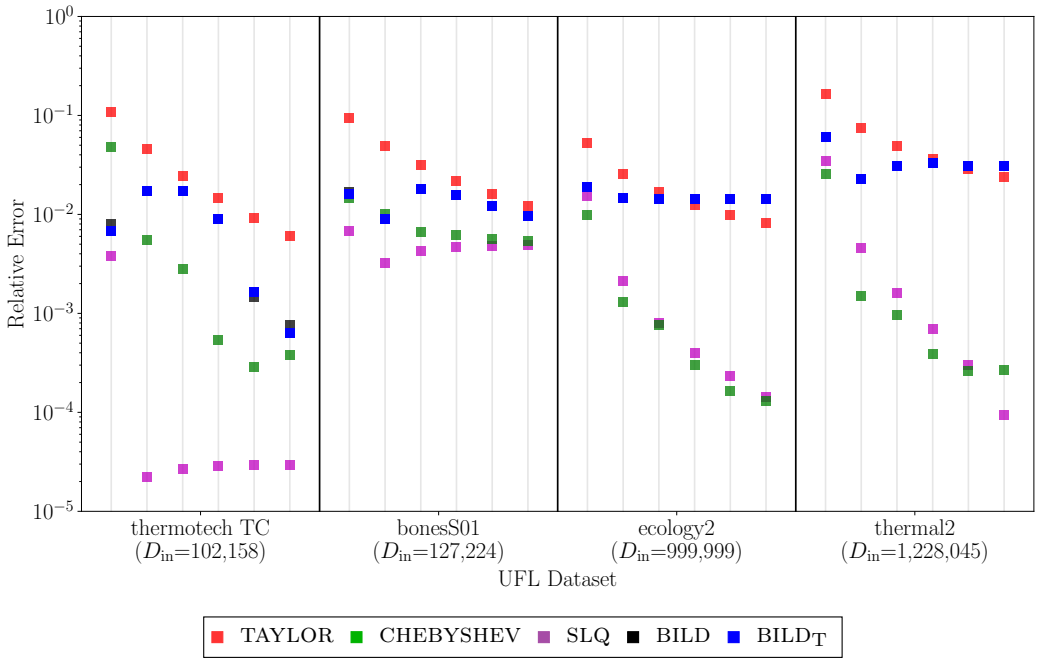


Fig. 4.2 Log determinant approximations compared on a variety of UFL Sparse datasets. For each dataset, the matrix was raised up to powers of 5, 10, 15, 20, 25, and 30 (left to right) using STE. This corresponds to the number of MVMs carried out by each method.

up an experiment whereby a selection of matrices with varying eigenspectra are synthetically constructed. Each matrix is built using a Gaussian kernel evaluated over 1,000 input points. Additionally, we verify the performance of each technique when assigned an increasing allowance of matrix vector multiplications (MVMs), which roughly corresponds to the computational budget allocated to each method.

The outcome of this experiment is visualised in Figure 4.1. The top half of this figure illustrates the eigenvalue decay rates for the six constructed matrices, where the decay rate becomes progressively larger. In the corresponding comparison between methods shown in the lower half of the figure, we can see that for matrices having slowly-decaying eigenvalues, the standard Taylor approximation fares quite poorly. The truncated version of our model, BILD$_{\text{T}}$, is particularly effective when the bounds are tighter, as evidenced for the EIGS-6 example having a rapid eigenvalue decay rate. On the downside, we must also point out that the performance does not seem to be greatly affected by the allowance of MVMs; as we shall elaborate further on, this is not ideal as we would prefer the performance to consistently improve when the method is assigned a greater computational budget.

4.3.2 UFL Sparse Datasets

The method we proposed is amenable to any positive semi-definite matrix whose eigenvalues are normalised to lie between 0 and 1. To this end, we extend the previous experimental set-up to a selection of real, sparse matrices obtained from the SuiteSparse Matrix Collection (Davis and Hu, 2011). Note that the dimensionality of the largest matrix is over a million, for which computing the exact log determinant using the Cholesky decomposition is intractable. Following Ubaru et al. (2017), we plot the relative error with respect to the true values of the log determinant reported in Boutsidis et al. (2017), and compare the three approaches to this baseline. The results for this experiment are shown in Figure 4.2.[1] Once again, the estimates obtained using our Bayesian approach achieve comparable accuracy to the Taylor approximation, but as in the previous experiment, we also observe that there is no guaranteed increase in the quality of the approximation when more MVMs are assigned. The benefit of incorporating bounds is also negligible in this set-up. Furthermore, as alluded to in Section 4.2.1, the Taylor approximation is not necessarily the best available means of estimating the log determinant of a large matrix. In this experiment, we highlight this point by also including results obtained by a more recent alternative approach relying on stochastic Lanczos quadrature (SLQ; Ubaru et al., 2017), and another using Chebyshev polynomial expansions (Han et al., 2015). The latter approaches yield superior results in nearly all reported problem settings, which accentuates the accuracy trade-off incurred by our proposal.

4.3.3 Uncertainty Quantification

Beyond its predictive performance, the defining feature of our proposal is its ability to quantify the uncertainty of the predicted log determinant, which can be interpreted as an indicator of approximation quality. This feature is unique to our approximation which to date, and to the best of our knowledge, remains the only Bayesian approach for estimating the log determinant of a large matrix. In order to verify the correctness of uncertainty estimates returned by our model, we reconsider the predictions evaluated for the previous experiment on datasets obtained from the SuiteSparse Matrix Collection. More specifically, we report the ratio of the absolute error to the predicted standard deviation; for the latter to be meaningful, or at least well-calibrated, we would expect the error to lie within only

[1] Due to an error in our original implementation of Chebyshev and SLQ for this comparison, an earlier version of this figure appearing in Fitzsimons et al. (2017a) understated the superior performance of the aforementioned techniques. On the contrary, the updated results shown in Figure 4.2 corroborate the improvements in performance expected of these methods.

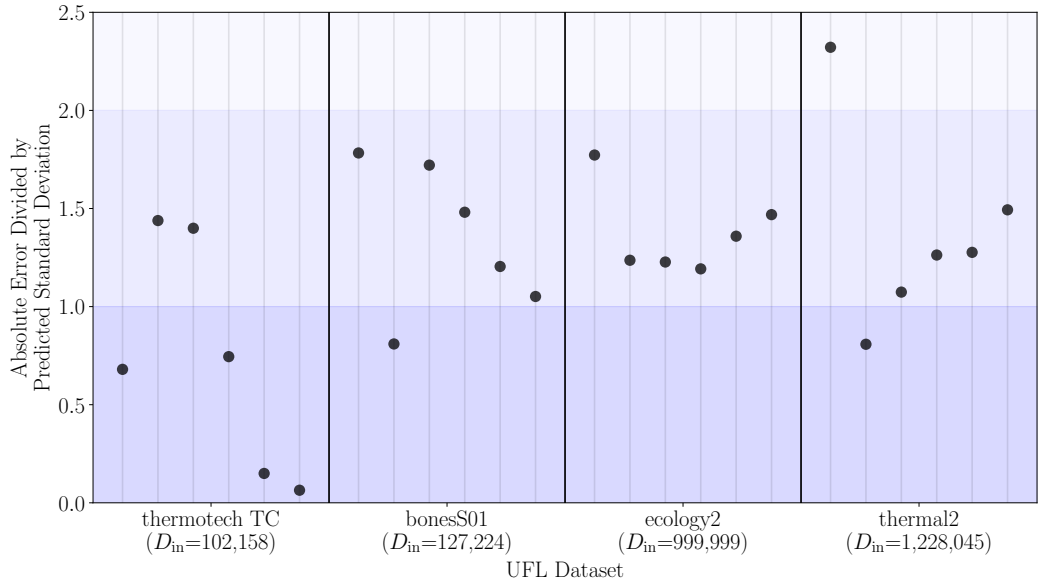


Fig. 4.3 Quality of uncertainty quantification of log determinant estimates on UFL datasets, measured as the ratio of the absolute error to the predicted standard deviation. As before, results are shown for increasing computational budgets (MVMs). The absolute error fell outside two standard deviations of the prediction in only one of 24 trials.

a few multiples of the standard deviation. As evidenced in Figure 4.3, the absolute error is bounded by at most twice the predicted standard deviation in all but one case. This corroborates the suitability of our method for not just yielding sensible point estimates of the log determinant, but rather a posterior distribution over candidate solutions. This is particularly pertinent to set-ups where this approximation appears as a component in a computational pipeline in which uncertainty must be effectively propagated from one step to the next.

4.3.4 Alternative Approximation using Maximum Entropy

Inspired by the probabilistic interpretation of estimating log determinants described in this chapter, in follow-up work (Fitzsimons et al., 2017b) we developed an alternative approximation that is instead rooted in information theory. In particular, we exploit the relationship between STE and the moments of a matrix's eigenspectrum by treating these estimates as moment constraints on the probability distribution of eigenvalues. This is achieved by maximising the entropy of the probability density $p(\lambda)$ with respect to the estimated moment constraints. Delving into the detail of this methodology is beyond the scope of this thesis; however, this formulation makes different prior assumptions to BILD

that enable it to perform better than the latter on selected benchmark examples. Nonetheless, that method does not yield uncertainty estimates, which was conversely the defining characteristic and primary motivation for developing BILD.

4.4 Linking Probabilistic Numerics to Gaussian Processes

In Chapter 3, we demonstrated how GP training and inference can be scaled to large datasets by approximating linear algebraic operations given some budget constraints. Although such schemes are effective for enabling tractability, they also introduce an additional source of computational uncertainty that is not accounted for in the uncertainties obtained from the model. In this section, we put forward our ideology that casting linear algebraic operations as inference problems is a natural strategy for sensibly quantifying the numerical uncertainty inherent to algebraic approximations used to accelerate 'exact' GPs. Complementary to our Bayesian approach for estimating log determinants, we start by outlining recent work on developing probabilistic algorithms for solving linear systems. We then combine both techniques to show how these are sufficient for characterising the uncertainty appearing in the algebra of GPs. Although this section is primarily intended to establish the foundations for future work, we conclude this chapter with a brief discussion on how the suite of tools currently available for carrying out probabilistic numerics may not yet be robust enough to unreservedly handle such tasks.

4.4.1 Probabilistic Linear Solvers

In the spirit of solving linear systems given a restricted computational budget, probabilistic linear solvers are intended to capture the reliability of an approximate, budget-constrained solution by returning a probability distribution over the approximation. More specifically, a probabilistic iterative linear solver should encode the reduction in uncertainty of approximate solutions obtained in subsequent iterations of the algorithm by reflecting the contraction in the span of possible solutions following each iteration.

A very recent paper covering and extending such methods (Bartels et al., 2018) distinguishes between two principal categories of inference strategies, namely *matrix*-based and *solution*-based inference. Foundational work presented in Hennig (2015) falls into the former category; in particular, given a linear system of the form $A\mathbf{z} = \mathbf{v}$, inference is carried out explicitly on A^{-1}, implying a prior and posterior distribution over all N^2 elements of the matrix. This requirement could very easily lead to intractability, which is why the

prior and resulting posterior are designed in such a way that cumbersome algebraic operations can be accelerated through the use of Kronecker algebra. Assuming a flattened representation of A^{-1} denoted by $\overrightarrow{A^{-1}}$, the proposed prior takes the form

$$p\left(\overrightarrow{A^{-1}}\right) = \mathcal{N}\left(\overrightarrow{A_0^{-1}}, \Sigma_0 \otimes W_0\right), \tag{4.34}$$

where Σ_0 and W_0 are intended to respectively capture the dependencies between columns and rows of A^{-1}. In this setting, the partial observations at iteration $t+1$ take the form of AS_t, where S_t is an $N \times t$ matrix constructed by horizontally stacking the t preceding search directions. The posterior over the solution to the linear system itself, $\mathbf{z}_\star$, can then be analytically derived using Gaussian identities.

On the other hand, and as the name implies, solution-based inference relates to procedures in which inference is carried out directly on the solution to the linear system, $\mathbf{z}_\star$. In this regard, the Bayesian conjugate gradient approach (BCG; Cockayne et al., 2018) requires a prior of the form

$$p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}_0, \Sigma_0\right). \tag{4.35}$$

Here $\mathbf{z}_0$ can be initialised to a vector of zeros, while Σ_0 denotes the prior covariance among entries of $\mathbf{z}$. In this set-up, the partial observations at each iteration now take the simplified form of $S_t^\top \mathbf{v}$. This approach is intrinsically more appealing given that the ultimate objective $\mathbf{z}_\star$ is being modelled directly. However, since the matrix-based approach yields a posterior solution to the intermediate term A^{-1}, there is greater scope for reusing this result in other computations beyond just the linear system being solved. In the context of GPs, this is particularly pertinent to computing the variance of predictions on test data, where a linear system involving the same matrix must be solved for every test point. In the remainder of this chapter tying to GPs, we limit our discussion to solving linear systems using either the matrix-based approach of Hennig (2015) or BCG; the Bayesian preconditioning scheme devised in Bartels et al. (2018) is also of direct interest to this work, but was published too recently to be properly considered in this thesis.

4.4.2 Application to Gaussian Processes

Consider the log marginal likelihood of a standard GP,

$$\log\left[p(\mathbf{y}|X,\boldsymbol{\theta})\right] = -\frac{1}{2}\log|K_\lambda| - \frac{1}{2}\mathbf{y}^\top K_\lambda^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi. \tag{4.36}$$

For very large N, we could approximate the computation of the log determinant and the linear system appearing in this expression using standard algebraic approximations. However, if we instead approximate these terms using the Bayesian approaches detailed in this chapter, we can obtain a distribution over the log marginal likelihood that accounts for the potential numerical uncertainty introduced by way of algebraic approximation. Assuming the approximated solutions for the log determinant $\log |K_\lambda|$ and quadratic form $\mathbf{y}^\top K_\lambda^{-1}\mathbf{y}$ to be $\mathcal{N}\left(\mu_{\mathrm{LD}}, \sigma^2_{\mathrm{LD}}\right)$ and $\mathcal{N}\left(\mu_{\mathrm{LS}}, \sigma^2_{\mathrm{LS}}\right)$ respectively, this gives us a probabilistic distribution over the log marginal likelihood that can be computed as

$$p\left(\mathcal{L}_{\mathrm{ML}}\right) = \mathcal{N}\left(-\frac{1}{2}\mu_{\mathrm{LD}} - \frac{1}{2}\mu_{\mathrm{LS}} - \frac{N}{2}\log 2\pi, \sigma^2_{\mathrm{LD}} + \sigma^2_{\mathrm{LS}}\right). \tag{4.37}$$

Similar equations can be derived for the gradients of $\mathcal{L}_{\mathrm{ML}}$, which as shown in Chapter 3 can be expressed exclusively in terms of linear systems that can be solved with a probabilistic linear solver. This also extends to inference with GPs, whereby predictions should also consider the numerical uncertainty introduced by approximating the solution of the required linear systems. Introducing μ_{LSM} and σ^2_{LSM} to denote the mean and variance of solving $\mathbf{k}_\star^\top K_\lambda^{-1}\mathbf{y}$, as well as μ_{LSV} and σ^2_{LSV} for $\mathbf{k}_\star^\top K_\lambda^{-1}\mathbf{k}_\star$, for a single test point $\mathbf{x}_\star$ we obtain

$$y_\star \sim \mathcal{N}\left(\mu_{\mathrm{LSM}}, k_{\star\star} - \mu_{\mathrm{LSV}} + \sigma^2_{\mathrm{LSV}} + \sigma^2_{\mathrm{LSM}} + \lambda\right), \tag{4.38}$$

where the key quantity of interest is the inflated variance including numerical uncertainty.

Although formulating these expressions as probability distributions is not the norm in existing GP methodologies, we foresee several applications where such a shift in perspective may be warranted:

- **Outlier and Change-point Detection**

  Well-calibrated uncertainty estimation is essential when GPs are applied to problems such as novelty and anomaly detection, where action is taken when the uncertainty for a prediction exceeds a predetermined threshold. This is particularly pertinent to engineering applications such as vehicular design, for which domain-specific quantification of computational uncertainty has been considered in various forms (Petrone, 2011). Precisely quantifying all possible sources of uncertainty is also highly sought after in financial modelling and time-series forecasting (Roberts et al., 2013), where risk should never be underestimated.

- **Multi-fidelity Bayesian Optimisation**

Wu and Frazier (2017) propose a multi-fidelity Bayesian optimisation procedure, based on earlier work (Wu et al., 2017) incorporating gradient information in Bayesian optimisation procedures, that additionally handles potentially noisy objective functions and derivatives. However, in spite of the framework's capacity to handle uncertain quantities, the authors struggle to give practical examples where this may arise, and consequently restrict their analysis to problems with synthetically injected noise. On the other hand, if we consider the formulation of the GP marginal likelihood given in Equation 4.37, along with its similarly uncertain gradients, this serves as an interesting case study in which the fidelity measure corresponds to the computational budget assigned to evaluating the objective function (in this case the log marginal likelihood) and the uncertainty tied to its evaluation.

- **Pipelined Decision Making**

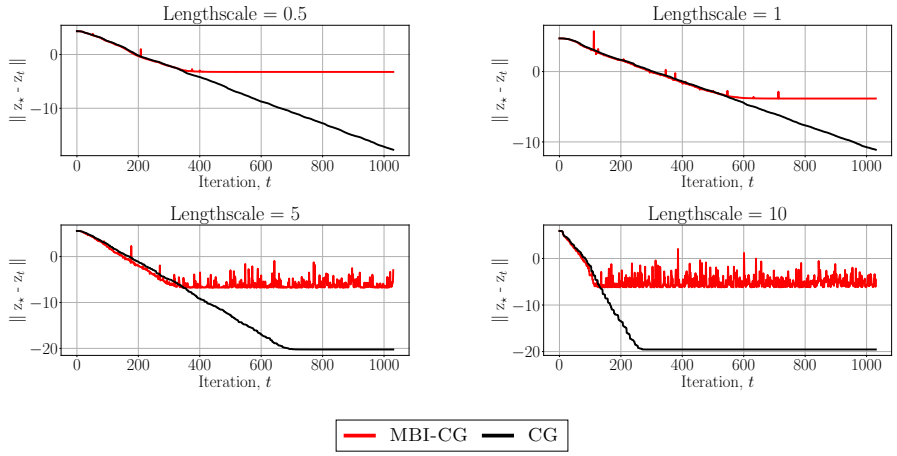
  As highlighted by Hennig et al. (2015) in their comprehensive introduction to probabilistic numerical methods, quantifying the uncertainty associated with approximate computation is particularly important when GPs appear in an iterative procedure or computational pipeline. One such application is experimental design (Morris, 2004), whereby correctly evaluating the underlying GP model's uncertainty is crucial for selecting sample points at each iteration which best exploit the exploration-exploitation trade-off.

As an aside, it is worth mentioning that advances in the quantification of numerical uncertainty must be complemented by also developing more flexible frameworks for effectively ingesting this additional information and adapting computation accordingly.

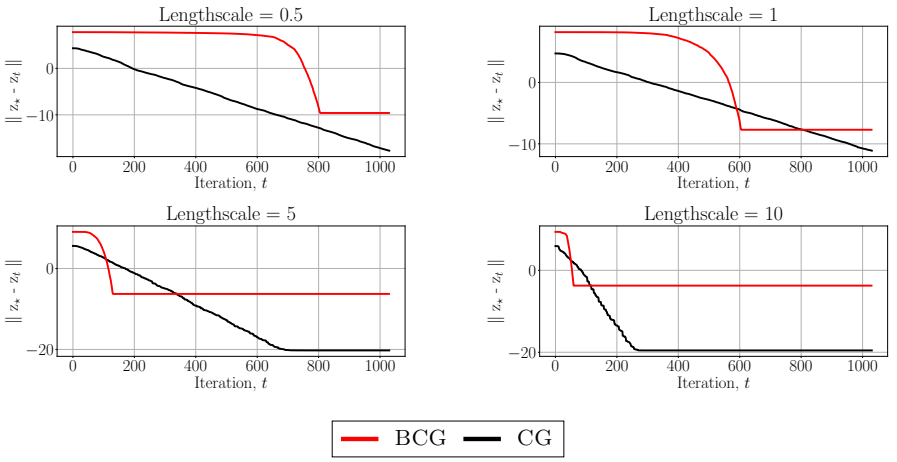
4.4.3 Beyond Theoretic Appeal - A Cautionary Note

The aforementioned use-cases are indicative of the applications where GPs augmented with probabilistic numerics can be put to good use. However, the success of such schemes depends entirely on the quality of the underlying Bayesian algebraic approximations. If either the reliability of predictions is inconsistent or the associated numerical uncertainty is badly calibrated, then incorporating probabilistic numerics is more likely to thwart a decision making procedure than enhance it.

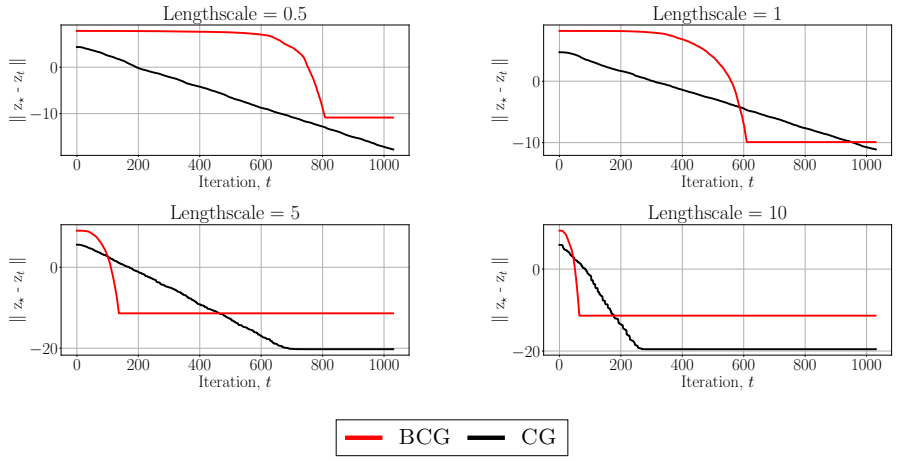
Although probabilistic numerical methods have been shown to perform adeptly on well-conditioned, predominantly sparse matrices, their application to GPs presents two major challenges. Firstly, in order to capture sensible correlation between data, kernel



(a) Matrix-based inference from Hennig (2015).

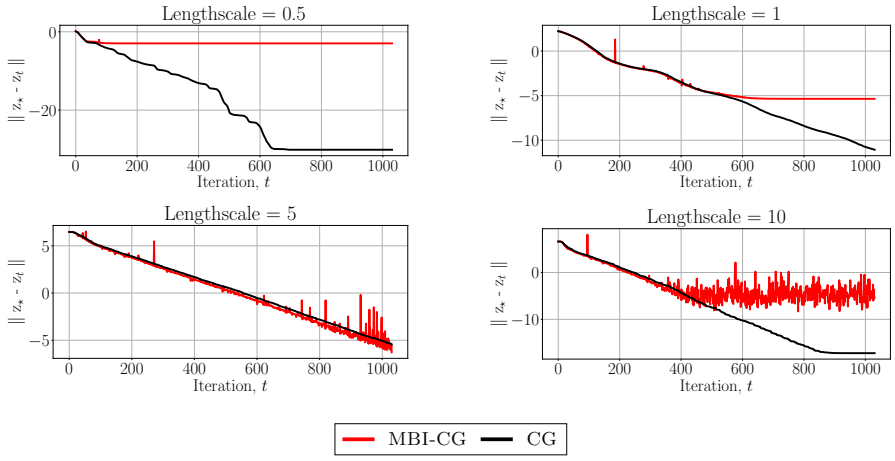


(b) Solution-based inference from Cockayne et al. (2018) with prior $\Sigma_0 = I_N$.

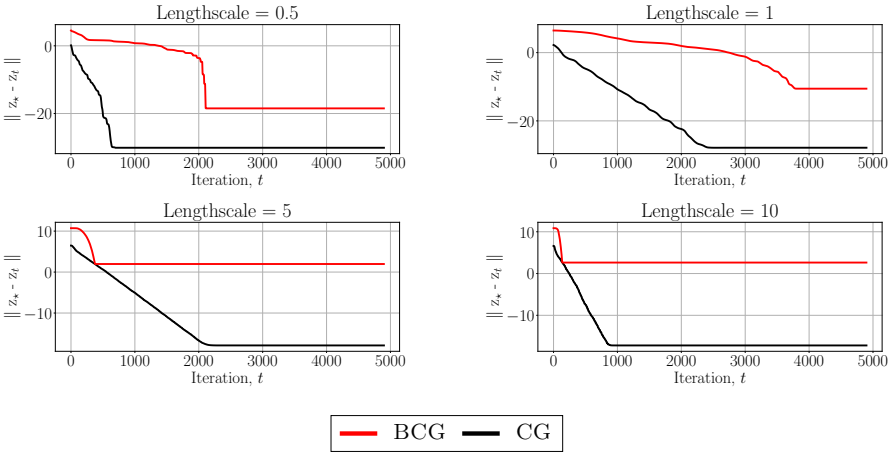


(c) Solution-based inference from Cockayne et al. (2018) with prior $\Sigma_0 = P^{-1}$.

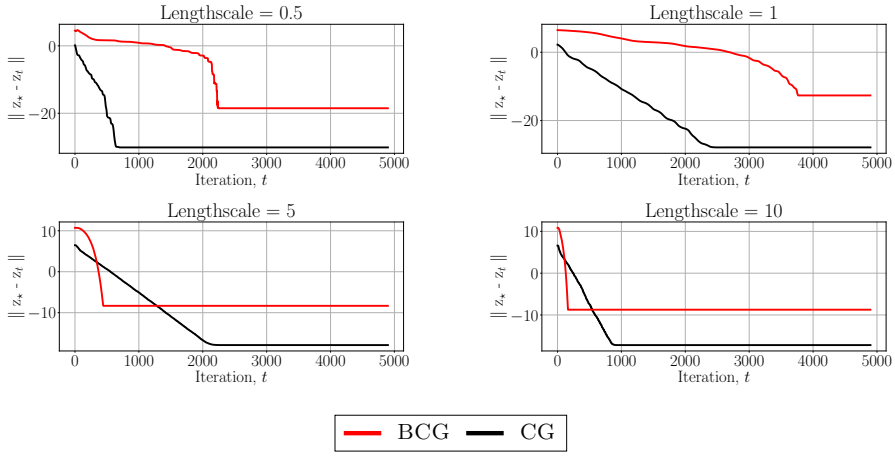
Fig. 4.4 Evaluation of probabilistic linear solvers for kernel matrices. The plots show the norm of the error (in log scale) at every step of the iterative algorithm. The evaluation is carried out for the **Concrete** dataset ($N = 1030$, $D_{\text{in}} = 8$).



(a) Matrix-based inference from Hennig (2015).

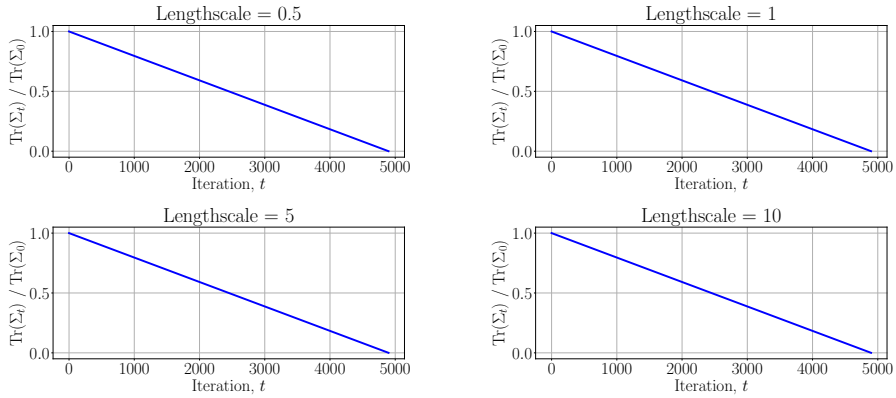


(b) Solution-based inference from Cockayne et al. (2018) with prior $\Sigma_0 = I_N$.

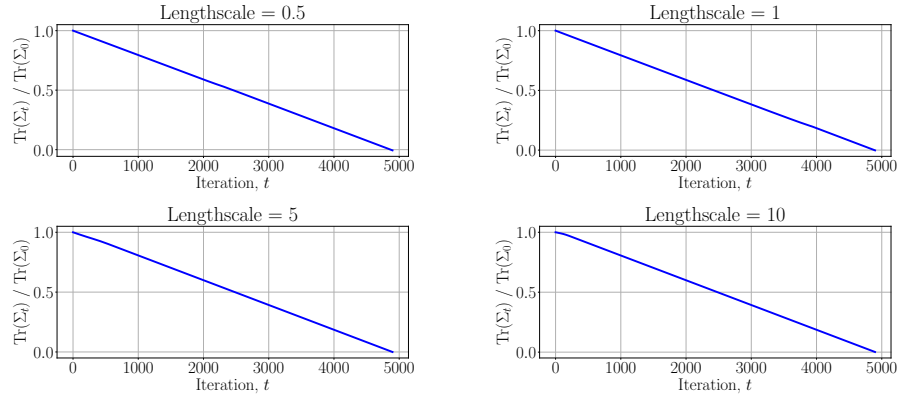


(c) Solution-based inference from Cockayne et al. (2018) with prior $\Sigma_0 = P^{-1}$.

Fig. 4.5 Evaluation of probabilistic linear solvers for kernel matrices. The plots show the norm of the error (in log scale) at every step of the iterative algorithm. The evaluation is carried out for the **White Wine** dataset ($N = 4898$, $D_{\text{in}} = 11$).



(a) Uncertainty quality of solution-based inference with prior $\Sigma_0 = I_N$.



(b) Uncertainty quality of solution-based inference with prior $\Sigma_0 = P^{-1}$.

Fig. 4.6 Evaluation of uncertainty obtained from probabilistic linear solvers for kernel matrices evaluated over the **White Wine** dataset. We plot the summation of variances across the diagonal of the posterior covariance, $\mathrm{Tr}\left(\Sigma_t\right)$, normalised by the trace of the prior covariance I_N or P^{-1}. This is repeated for the **Concrete** dataset in Appendix B, yielding similar results.

matrices are often dense and likely to be badly conditioned. The iterative nature of GP parameter optimisation necessitates that these algorithms must be applied to a different kernel matrix or linear system at every iteration, and probabilistic numerical methods can be particularly unreliable when the kernel is constructed using extreme parameter values. Secondly, although these methods are generally evaluated over small matrices with dimensionality in the order of hundreds, the notion of carrying out GP inference on a budget is mostly pertinent to datasets having thousands or even millions of data-points. In the remainder of this section, we shift our discussion to the first requirement, and illustrate how state-of-the-art probabilistic numerical methods currently perform in this regard.

In our evaluation, we consider the Concrete dataset (N=1030, D_{in}=8) and the White Wine dataset (N=4898, D_{in}=11) from the UCI dataset repository (Asuncion and Newman, 2007), and compare the aforementioned probabilistic linear solvers to regular CG for a selection of kernel constructions. More specifically, we vary the lengthscale of the RBF covariance in the range $[0.5, 1, 5, 10]$ and plot the norm of the error (in log scale) at every step of the iterative algorithm. We set the variance to one, and add diagonal noise (fixed at 0.001) to the resulting covariance matrices. The linear systems to be solved are of the form $K_\lambda \mathbf{z} = \mathbf{y}$.

For BCG we investigate two priors: one is simply $\mathcal{N}\left(\mathbf{0}, I_N\right)$, while the other is a preconditioned prior of the form $\mathcal{N}\left(\mathbf{0}, P^{-1}\right)$, where we set P to be a Nyström preconditioner constructed with $M=\sqrt{N}$ inducing points. Given that convergence for the linear system solved by BCG is bounded by the condition number $\kappa\left(K_\lambda^T K_\lambda\right)$, Cockayne et al. (2018) suggest a preconditioner prior covariance having the form $\left(P^T P\right)^{-1}$. However, in our evaluation we observed that the results were more stable when Σ_0 is set to P^{-1}; this more closely resembles the hypothetical $\Sigma_0 = K_\lambda^{-1}$ prior explored in that same work.

The results are plotted in Figures 4.4 and 4.5, from which it can be immediately observed that the matrix-based inference approach fails to converge as well as CG, and instead plateaus at inferior optimal solutions. Our personal implementation of the algorithm is also sometimes prone to instability, as witnessed for the set-up with the kernel having the largest lengthscale. On the contrary, both variations of BCG consistently converge to the optimal solution at a rate that is even sometimes faster than regular CG. Nonetheless, using the implementation provided by the authors[2], in our evaluation we observed that numerical instability could result in the algorithm breaking down in multiple instances. In our evaluation, we compensate for this issue by only plotting the results up until the instability occurs, after which we report the optimal result up to that iteration until termination. This issue occurs despite using batch-normalised search directions, which are expected to improve the stability of the approximation.

Meanwhile, in Figure 4.6 we verify the quality of the uncertainty estimates obtained from BCG for the same executions illustrated in Figures 4.5b and 4.5c respectively (the evaluation is repeated for the Concrete dataset in Appendix B). In particular, we plot the summation of variances along the diagonal of the posterior covariance, $\text{Tr}\left(\Sigma_t\right)$, normalised by the trace of the prior covariance I_N or P^{-1}. Disappointingly, the decrease in uncertainty is exactly linear when the prior covariance is set as the identity matrix, and very little change is observed when this is updated to P^{-1}. Since the reduction of uncertainty is

[2]https://github.com/jcockayne/bcg

directly proportional to the number of iterations elapsed, such uncertainty yields very little (if any) additional information in a decision making procedure. These findings corroborate the similarly wary analysis provided in Bartels and Hennig (2016) and Bartels et al. (2018) regarding the uncertainty of probabilistic linear solvers.

Similar problems were also observed for BILD, whereby the log determinant approximation tends be very inaccurate when the matrix is badly conditioned. This can be partly attributed to the Taylor approximation of the log determinant upon which BILD is based, since similar behaviour was also observed for that technique in our comparison of log determinant approximations. This problem also extends to the quantification of uncertainty, which can at times be orders of magnitude larger than the error. Of greater concern, unlike the results shown for probabilistic solvers in Figures 4.4, 4.5 and 4.6, the quality of the prediction and associated uncertainty does not always improve when allocated a greater computational budget (this is made apparent in Figures 4.2 and 4.3).

At a glance, this discussion paints a tepid outlook for the widespread application of probabilistic numerics to kernel-based learning models. However, this is an evolving field that has attracted great interest and attention in recent years, and we envisage that ongoing effort to improve the performance and robustness of fundamental probabilistic numerical algorithms could lead to more promising outcomes in the near future. It is also worth noting that in this chapter, we have restricted our discussion to procedures falling directly under the umbrella of Bayesian probabilistic numerics; however, future work in the direction of quantifying the reliability of algebraic approximations could widen the scope of this discussion to more general probabilistic interpretations of linear algebra to similar effect.

4.5 Conclusion

We opened this chapter by describing a novel Bayesian framework for approximating log determinants. In particular, our approach enables the log determinant of large matrices to be inferred from noisy observations of raw moments obtained using stochastic trace estimation. By modelling the underlying eigenvalue distribution using a GP, a posterior estimate for the log determinant can then be computed using Bayesian quadrature. Our experiments indicate that the results are not always directly comparable to state-of-the-art methods, but our proposal instead enables the quantification of uncertainty associated with the approximation. More significantly, this work contributes towards a larger shift in perspective of reinterpreting linear algebra as inference problems, with particular empha-

sis on their application to Gaussian processes. However, although probabilistic numerical methods are theoretically appealing and indeed work well for a variety of problem settings, this is still a developing field of study. As discussed in the remainder of the chapter, it is difficult to tackle more applied problems involving probabilistic numerics until the foundations are made more robust. Nonetheless, our work remains an important step forward in the direction of fully characterising the computational uncertainty associated with approximating large-scale kernel-based methods.

Chapter 5

Bridging the Gap between Gaussian Processes and Deep Learning

Our discussion on scalable Gaussian processes has thus far been limited to comparisons against other GP approximations for medium-sized datasets. However, real-world data poses a greater challenge with regards to requiring both superior modelling flexibility and also better scalability to truly large datasets. In this chapter, we broaden the scope of our evaluation to consider the role and significance of GPs in the context of more widely-used deep learning methods. Building upon the content presented thus far, we now shift our attention to how GP training and inference can be adapted to the big data regime. We start by presenting AutoGP, a model developed in collaboration with Krauth et al. (2017) having the primary intent of fully exploring the capabilities and limitations of GPs with application to traditional deep learning problems. Although such approaches have made great strides in the direction of 'modernising' GPs, deep Gaussian processes (DGPs) are a more natural candidate for emulating the behaviour of widespread deep learning techniques. Based on the work presented in Cutajar et al. (2017), in this chapter we introduce a novel formulation of DGPs based on random feature expansions that we train using stochastic variational inference. This yields a practical learning framework which significantly advances the state-of-the-art in inference for DGPs, while also enabling accurate quantification of uncertainty.

5.1 Overview

The GP approximations considered so far in this thesis are well-suited to datasets with tens of thousands of observations, but may not be ideal when millions or even billions of datapoints are available. Similarly, standard GP kernels such as RBF and Matérn are insufficient for capturing the complex covariance between many real-world observations, motivating the development of GP models that allow for the implicit construction and composition of more complex kernels. The models presented in this chapter are intended to embody these desiderata.

Stochastic variational inference for GPs (Hensman et al., 2013), which we covered in Section 2.2.3 of Chapter 2, yields an immediate means for scaling computation to truly large datasets. In particular, assuming a likelihood that is fully-factorised over observations, and using a global variational approximation of the inducing points, the GP model can be trained using mini-batch-based stochastic optimisation. The largest dataset considered in its original presentation had 800,000 observations, but the analysis was limited to regression problems. This was later extended to classification problems in Hensman et al. (2015b), where the performance of the proposed contribution was evaluated by reinterpreting the large-scale experiment in the aforementioned work as a binary classification problem. Multi-class classification problems such as MNIST (LeCun and Cortes, 2010) were also investigated in that work, with the obtained results being considered state-of-the-art for GPs at the time, albeit using a standard RBF kernel with a single lengthscale parameter shared across all input dimensions. Follow-up work in Hensman et al. (2015a) further improved the performance of this approach by introducing a non-Gaussian variational approximation that utilises Markov chain Monte Carlo (MCMC) for sampling from the variational posterior.

An alternative category of approximations relies on treating the conditional likelihood as a black-box function (Dezfouli and Bonilla, 2015; Ranganath et al., 2014). This implies that detailed knowledge of its implementation or gradients are not required as long as they can be sampled efficiently, and is tied to an underlying preference for variational approximations over MCMC sampling; this reinforces the assertion that optimisation is an easier problem than integration. Whereas variational approximations typically necessitate model-specific formulations, the use of black-box likelihoods widely broadens the generality of such methods. Bonilla et al. (2016) demonstrate the effectiveness of this approach on both multi-output regression tasks with complex likelihoods, such as Gaussian process regression networks (Wilson et al., 2012), as well as large-scale classification problems.

The application of GPs to large datasets must also be complemented with kernels that can capture more complex interactions between data-points. Several independent works have addressed this requirement, which include the introduction of spectral mixture kernels (Wilson and Adams, 2013), arc-cosine kernels emulating the computation of multi-layer neural networks (Cho and Saul, 2009), and convolutional kernels (Mairal et al., 2014). Of particular interest has been the renewed popularity of deep kernel learning (Wilson et al., 2016a), whereby the inputs to a kernel are first transformed using a deterministic or non-deterministic warping scheme. Key applications of deep kernel learning in image classification tasks employ convolutional neural networks (CNNs; LeCun et al., 2015) in order to obtain a sensible transformation of the data that can then be pipelined as input to a standard GP (or any GP approximation). This was also recently applied to time series data (Al-Shedivat et al., 2017), whereby an LSTM is used to obtain a deterministic transformation of the inputs that is able to capture long term dependencies between observations. In order to remain tractable when applied to large datasets, a variation referred to as stochastic variational deep kernel learning (SV-DKL; Wilson et al., 2016b) was introduced for scaling up such architectures to datasets having millions of observations. Apart from enabling the use of mini-batches, the scalability of this approach also relies heavily on the Kronecker-based structured kernel interpolation scheme discussed in Section 2.4 of Chapter 2.

All of the aforementioned works are unified by the overarching goal of improving the performance of GPs to match the results obtained by deep learning techniques. In this chapter, we first present AutoGP - a state-of-the-art GP model intended to directly compete with the benchmark results obtained by deep learning techniques on both regression and classification tasks. Jointly addressing aspects such as automated variational inference and employing a leave-one-out-based objective function for hyperparameter learning results in an augmented GP model that successfully emulates the primary achievements of deep learning techniques. AutoGP outperforms all preceding GP models on the MNIST benchmark, and its noteworthy scalability is showcased by applying it to the MNIST-8M dataset (Loosli et al., 2007), which artificially extends the MNIST dataset to 8.1 million training points by pseudo-randomly transforming the original training set of 60,000 images.

Although AutoGP makes great strides towards bridging the gap between GPs and deep learning techniques, the compositional structure of deep Gaussian processes (DGPs; Damianou and Lawrence, 2013) presents a more intuitive approach to achieve this goal. More generally, the composition of multiple GPs as a DGP enables a deep probabilistic non-

parametric approach to flexibly tackle complex machine learning problems with sound quantification of uncertainty. Because of their probabilistic formulation, it is reasonable to approach the learning of DGPs through Bayesian inference; however, the application of such techniques to learn DGPs leads to various forms of intractability. A number of contributions have been proposed to recover tractability (Bui et al., 2016; Dai et al., 2016; Hensman and Lawrence, 2014), but even among these works, there does not seem to be a singular approach that is truly applicable to large-scale problems and tractable beyond just a handful of hidden layers.

In this light, we develop a practical learning framework for DGP models that significantly improves the state-of-the-art on those aspects. In particular, our proposal introduces two sources of approximation to recover tractability, while (i) scaling to large-scale problems; (ii) being able to work with moderately deep architectures, and (iii) accurately quantifying uncertainty. The first is a model approximation whereby the GPs at all layers are approximated using random feature expansions (Lázaro-Gredilla et al., 2010; Rahimi and Recht, 2008), while the second approximation relies upon stochastic variational inference to retain a probabilistic and scalable treatment of the approximate DGP model. In this chapter, we demonstrate how the resulting DGP model can handle significantly more layers than are usually assumed within the literature on DGPs, while also being scalable to large datasets for which DGP inference had previously been computationally infeasible. More significantly, we show that the model consistently outperforms competing techniques both in terms of convergence speed and predictive performance.

5.2 Exploring the Capabilities and Limitations of Gaussian Processes with AutoGP

Given their impressive performance on machine learning and pattern recognition tasks, deep learning techniques have attracted a considerable deal of attention in several applied domains such as computer vision and natural processing; see e.g., LeCun et al. (2015) and references therein. Application-specific kernels such as those presented in Mairal et al. (2014) indicate that kernel-based methods can be successfully adapted to deliver comparable results to established deep learning techniques, but the jury is still out as to whether GPs can more generally compete with their flexibility and superior predictive performance. In this section, we report on this goal by summarising the primary contributions and results obtained by AutoGP, a model developed for the purpose of investigating three complemen-

tary directions for improving the performance of GPs, namely (i) scalable and statistically efficient inference; (ii) flexible kernels; and (iii) objective functions for hyperparameter learning alternative to the marginal likelihood. Given that the centrepiece of this chapter will be our work on developing scalable DGPs, we limit the discussion of this work in this thesis to briefly explaining the primary outcomes of this study, with particular emphasis on the aspects shared with the DGP approximation.

5.2.1 Automated Variational Inference

The first aspect targeted by AutoGP is the quality of the approximate GP posterior, where we build our construction upon the variational free energy model (VFE) presented in Titsias (2009) (see Section 2.2.2 of Chapter 2 for more details), where the approximate joint posterior is given by

$$q(\mathbf{f}|\mathbf{u}) = p(\mathbf{f}|\mathbf{u})\, q(\mathbf{u}). \tag{5.1}$$

As before, $\mathbf{u}$ denotes the M inducing points for introducing sparsity in the model. In order to cater for multi-output regression or multi-class classification, we extend the original presentation of $q(\mathbf{u})$ to introduce Q latent functions in the GP, such that

$$q(\mathbf{u}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{Q} \mathcal{N}\left(\mu_{\cdot j}; \mathbf{m}_{kj}, S_{kj}\right). \tag{5.2}$$

The number of latent functions, Q, is typically set to match the output dimensionality of the data, D_{out}. In the above, $\mathbf{m}$ and S denote the parameters of the $K \times Q$ variational distributions to be learned during the optimisation procedure. In a slight variation on the standard definition of $q(\mathbf{u})$, the variational posterior is reinterpreted as a mixture-of-Gaussians weighted by π_k, which is intended to introduce greater flexibility in the posterior. Meanwhile, the conditional $p(\mathbf{f}|\mathbf{u})$ is defined as

$$p(\mathbf{f}|\mathbf{u}) = \prod_{j=1}^{Q} \mathcal{N}\left(\mathbf{f}_{\cdot j}; \widetilde{\boldsymbol{\mu}}_j, \widetilde{K}_j\right), \tag{5.3}$$

where

$$\widetilde{\boldsymbol{\mu}}_j = A_j \mathbf{u}_{\cdot j}, \quad \text{and} \tag{5.4}$$

$$\widetilde{K}_j = K_{\text{XX}}^{(j)} - A_j K_{\text{UX}}^{(j)}, \tag{5.5}$$

with $A_j = \boldsymbol{K}_{\mathrm{XU}}^{(j)} \left(\boldsymbol{K}_{\mathrm{UU}}^{(j)} \right)^{-1}$.

In a similar procedure to that described for VFE in Chapter 2, the variational mixture-of-Gaussians posterior yields an evidence lower bound, $\mathcal{L}_{\mathrm{ELBO}}$, on the true marginal likelihood,

$$\mathcal{L}_{\mathrm{ELBO}} = \sum_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathbb{E}_{q_{k(i)}(\mathbf{f}_{i\cdot})} \left[\log p \left(y_i | \mathbf{f}_{i\cdot} \right) \right] - D_{\mathrm{KL}} \left[q \left(\mathbf{u} \right) || p \left(\mathbf{u} \right) \right], \tag{5.6}$$

where $D_{\mathrm{KL}} \left[q(\cdot) \,||\, p(\cdot) \right]$ once again denotes the Kullback-Leibler divergence between two distributions.

The Reparameterisation Trick

Computing $\mathcal{L}_{\mathrm{ELBO}}$ involves estimating the expectation appearing in the likelihood term; hence, the gradients of this bound must also be estimated when optimising model parameters. Approaches that assume black-box likelihoods are known to yield estimates of the gradients having very high variance; this could negatively impact the convergence rate of the optimisation procedure unless a very large number of samples is used, which itself incurs a different penalty on performance. This problem is exacerbated when complex likelihoods such as Gaussian process regression networks (Wilson et al., 2012) are employed.

A landmark paper by Kingma and Welling (2014) addresses this issue through the use of a so-called *reparameterisation trick* that can significantly clamp variance in the estimation of gradients. Given a random variable x conditioned on some parameter θ, this trick allows for rewriting x as a function of another random variable, ε, that is itself not conditioned on θ. Taking x to be a random variable drawn from $\mathcal{N}\left(\mu, \sigma^2\right)$, this implies rewriting x as

$$x = \mu + \varepsilon\sigma, \tag{5.7}$$

with $\varepsilon \sim \mathcal{N}\left(0, 1\right)$. In doing so, the gradients may be estimated by simply drawing univariate Gaussian samples in a Monte Carlo (MC) procedure. This formulation permits the use of automatic differentiation techniques (Baydin et al., 2017), which lifts the requirement of having to manually implement derivates for the target objective function.

In our set-up, this entails that the individual expectations in Equation 5.6 can each be estimated by taking samples (indexed by s) of

$$f_{ij}^{k,s} = b_{kij} + \varepsilon_{kij}^{s} \sigma_{kij}, \tag{5.8}$$

for $j = 1 \dots Q$, where every ϵ_{kij}^s is sampled from $\mathcal{N}(0, 1)$. In the above,

$$b_{kij} = \mathbf{a}_{ji}^\top \mathbf{m}_{kj}, \quad \text{and} \tag{5.9}$$

$$\sigma_{kij}^2 = \left[\widetilde{K}_j\right]_{i,i} + \mathbf{a}_{ji}^\top S_{kj} \mathbf{a}_{ji}, \tag{5.10}$$

where $\mathbf{a}_{ji}^\top = \left[A_j\right]_{:,i}$ denotes the M-dimensional vector corresponding to the i^{th} column of A_j, while $\left[\widetilde{K}_j\right]_{i,i}$ denotes the i^{th} diagonal entry of $\widetilde{K}_j$. Denoting the number of MC samples as $N_{\mathbf{s}}$, the estimated likelihood of the i^{th} sample can be expressed as

$$\mathcal{L}_{\text{ELL}}^i = \frac{1}{N_{\mathbf{s}}} \sum_{k=1}^{K} \pi_k \sum_{s=1}^{N_{\mathbf{s}}} \log p\left(\mathbf{y}_i | \mathbf{f}_i^{k,s}\right), \tag{5.11}$$

which is amenable to mini-batch-based optimisation. Another key property of AutoGP is that contrary to other models using the reparameterisation trick (Dai et al., 2016; Kingma and Welling, 2014), the posterior over latent functions remains fully correlated. This is given by

$$q(\mathbf{f}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{Q} \mathcal{N}\left(\mathbf{f}_{\cdot j}; \mathbf{b}_{kj}, \mathbf{\Sigma}_{kj}\right), \tag{5.12}$$

where

$$\mathbf{b}_{kj} = A_j \mathbf{m}_{kj}, \quad \text{and} \tag{5.13}$$

$$\mathbf{\Sigma}_{kj} = \widetilde{K}_j + A_j S_{kj} A_j^\top. \tag{5.14}$$

Following Dezfouli and Bonilla (2015), this entails that the reparameterisation trick can be exploited for estimating $\mathcal{L}_{\text{ELBO}}$ using univariate samples while still retaining a full approximate posterior. This had previously only been investigated for GP models having a single latent function and corresponding Gaussian posterior (Nickisch and Rasmussen, 2008; Opper and Archambeau, 2009).

5.2.2 Flexible Kernel Design

Although kernel design is instrumental to the usefulness of GPs when applied to real-world problems, many practitioners often default to using a standard isotropic RBF kernel with heavy smoothness assumptions. The use of automatic relevance determination (ARD; see

Chapter 1) by assigning a separate lengthscale to each dimension can trivially increase flexibility in a straightforward manner. Even so, the implied computational overhead entails that such a scheme is not commonly applied to large datasets possibly having an equally large input dimensionality. Conversely, the possibility of using automatic differentiation within AutoGP allows for ARD to be exploited with only minor computational overhead.

A more general and expressive set of covariance functions is the family of arc-cosine kernels, which are recursively constructed with the primary intention of emulating the compositional structure of a deep neural network. An arc-cosine kernel of degree d and indexed by depth l is recursively defined as

$$k_d^{(l+1)}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{1}{\pi}\left[k_d^{(l)}\left(\mathbf{x}_i, \mathbf{x}_i\right) k_d^{(l)}\left(\mathbf{x}_j, \mathbf{x}_j\right)\right]^{d/2} J_d\left(\phi_d^{(l)}\right), \tag{5.15}$$

where the kernel at depth $l = 1$ is $k_d^{(1)}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{1}{\pi}\|\mathbf{x}_i\|^d \|\mathbf{x}_j\|^d J_d\left(\phi\right)$, and the angle corresponding to degree d and depth l is given by

$$\phi_d^{(l)} = \cos^{-1}\left(k_d^{(l)}\left(\mathbf{x}_i, \mathbf{x}_j\right)\left(k_d^{(l)}\left(\mathbf{x}_i, \mathbf{x}_i\right) k_d^{(l)}\left(\mathbf{x}_j, \mathbf{x}_j\right)\right)^{-1/2}\right). \tag{5.16}$$

All of the angular dependencies are then modelled with the function

$$J_d\left(\phi\right) = (-1)^d \left(\sin\phi\right)^{2d+1}\left(\frac{1}{\sin\phi}\frac{\partial}{\partial\phi}\right)^d\left(\frac{\pi - \phi}{\sin\phi}\right), \tag{5.17}$$

with the base angle defined as $\phi = \cos^{-1}\left(\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)$. Later on, in our discussion on DGP approximations, we will show that the first order arc-cosine kernel can also be handily approximated using rectified linear units.

5.2.3 Leave-One-Out Learning

The final aspect of GP modelling considered in our assessment is whether the marginal likelihood of the model is always the best criteria for model selection. From a Bayesian standpoint, the marginal likelihood is a sensible objective function as it implicitly enforces the principle of Occam's razor that we highlighted in the introduction to this thesis. However, although this is a reasonable choice in most cases, there may be instances where other alternative objective functions may be more appropriate. In their discussion on hyperparameter optimisation for GPs, Rasmussen and Williams (2006) briefly note that the leave-one-out (LOO) cross-validation objective function is particularly better suited when the model is misspecified (Wahba, 1990). The appeal of using a LOO objective is

also pertinent to classification problems where lower error rates and well-calibrated predictive probabilities are especially prized. Several works have considered replacing the marginal likelihood with a LOO cross-validation objective, including Sundararajan and Keerthi (2001), Sundararajan and Keerthi (2012) and Vehtari et al. (2016). However, these works generally struggle to make a strong enough case for selecting LOO learning over standard approaches, and the proposed methodologies do not scale to very large datasets.

To this end, we focus on the average leave-one-out predictive probability for hyperparameter learning in the proposed variational framework as an alternative to optimising the marginal likelihood. This objective function is obtained by excluding one data-point from the training set and computing its log predictive probability when training on the remaining data-points. By virtue of the formulation of LOO strategies, the final objective is then computed by taking an average of the results for all individual data-points. This naïvely implies training N different models, each time leaving out a single point; however, using the AutoGP framework, this can be computed without having to explicitly train N models. Instead, the leave-one-out objective function can be expressed as

$$\mathcal{L}_{\mathrm{LOO}}(\theta) \approx -\frac{1}{N}\sum_{i=1}^{N}\log\int\frac{q\left(\mathbf{f}_{i\cdot}|X,\mathbf{y},\theta\right)}{p\left(\mathbf{y}_i|\mathbf{f}_{i\cdot}\right)}\mathrm{d}\mathbf{f}_{i\cdot}, \tag{5.18}$$

where the aforementioned simplification stems from using the variational marginal posterior $q\left(\mathbf{f}_{i\cdot}|X,\mathbf{y},\theta\right)$ instead of the true marginal posterior $p\left(\mathbf{y}_i|\mathbf{f}_{i\cdot}\right)$ for the i^{th} data-point. We also note that we have made explicit the dependency of the posterior on all the data. As for the marginal likelihood term given in Equation 5.6, the individual expectations appearing in the expression above can also be estimated using MC sampling, while the additive structure once again permits the use of mini-batches.

Although this targets the kernel hyperparameters of the model, we are still required to optimise the remaining variational parameters appearing in the model approximation. This calls for an alternating optimisation scheme whereby we first estimate the approximate posterior $q\left(\mathbf{f}_{i\cdot}|X,\mathbf{y},\theta\right)$ through optimisation of $\mathcal{L}_{\mathrm{ELBO}}$, and then learn the hyperparameters via optimisation of $\mathcal{L}_{\mathrm{LOO}}$. Mini-batch-based stochastic gradient optimisation can be used in both cases, thus preserving the scalability expected of this model.

5.2.4 Summary of Results

All of the aforementioned aspects can be teased apart in order to assess their individual contribution towards improving GP performance, and this was indeed among the primary

motivations for the work undertaken in Krauth et al. (2017). However, given that this section is predominantly intended as a precursor to the presentation of deep Gaussian processes, we shall only briefly restate the principal results obtained by the AutoGP model combining all of the elements discussed above in comparison to deep learning benchmarks.

Arguably the most widely-used multi-class classification benchmark, the MNIST dataset (LeCun and Cortes, 2010) is a collection of images of handwritten digits with a fixed division of 60,000 training and 10,000 test points. Using 1,000 inducing points, an RBF kernel with ARD, as well as the alternating optimisation scheme mentioned above, AutoGP attains an error rate of 1.55% and a mean negative log likelihood (MNLL) of 0.061. These results outperform the work of Hensman et al. (2015a), Gal et al. (2014), and Bonilla et al. (2016), which respectively report optimal error rates of 1.96%, 5.95% and 2.74%. Prior to our work, these were previously considered to be the state-of-the-art results obtained using GPs for this dataset. Even so, these results are still not directly comparable to top-tier performance obtained by alternative models beyond simply GPs. Notably, at the time of the paper's publication, convolutional neural networks based on the architecture outlined in Simard et al. (2003) could already achieve an error rate of 1.19% without carrying out any pre-processing before training. Notwithstanding, this still shows that GPs can perform comparably to non-Bayesian methods with the additional benefit of also providing better-calibrated uncertainty estimates. As highlighted in the introduction to this thesis and also recent work (Kendall and Gal, 2017), constructing models that return well-calibrated posterior distributions in image classification tasks is imperative for their application to domains such as self-driving cars.

The MNIST-8M dataset (Loosli et al., 2007) artificially extends the MNIST dataset to 8.1 million training points, but retains the original test set. At the time AutoGP was introduced, and to the best of our knowledge, no other GP model had been applied to such a large multi-class classification problem. Using a similar set-up as described for MNIST, we report an error rate of 0.89% and an MNLL of 0.033, breaking the 1% error rate barrier for GP models. This confirms that AutoGP is able to perform competitively with deep architectures when some form of pre-processing is applied to the original dataset. Henao and Winther (2012) report similar performance on MNIST using an augmented active set and a 9^{th} degree polynomial kernel.

Application to more complex classification tasks inspires less confidence, however. The Rectangles-Image dataset is a binary classification task created for the explicit purpose of comparing shallow models and deep learning architectures, whereby the labels indicate whether a rectangle in the image has larger width or height. Here, AutoGP obtains a mini-

Table 5.1 Summary of optimal results obtained by AutoGP on benchmark classification tasks.

Dataset	Error Rate	MNLL
MNIST	1.55%	0.061
MNIST-8M	0.89%	0.033
Rectangles-Image	24.06%	0.485
CIFAR10	44.95%	1.333

mum error rate of 23.6% using 1,000 inducing points, which is well below the performance of deep architectures such as deep trans-layer autoencoder networks, 13.01% (Zhu et al., 2015), and invariant scattering convolutional networks, 8.01% (Bruna and Mallat, 2013). More discouragingly, we only obtain an error rate of 44.95% on CIFAR10, another commonly used image classification benchmark. This is a far cry from the error rates obtained by state-of-the-art deep learning techniques, which are frequently below 10%. The results obtained by AutoGP on the four benchmark datasets highlighted in this section are summarised in Table 5.1.

5.3 Deep Gaussian Processes

In Section 5.2.2, we commented on how deep kernels such as the arc-cosine kernel can be employed within standard GPs for the purpose of mimicking the computation of a deep neural network. However, this does not constitute proper function composition. On the other hand, the composition of multiple GPs as a single deep Gaussian process (DGP; Damianou and Lawrence, 2013) yields a deep probabilistic nonparametric model that more closely resembles the structure of other established deep learning models. From a generative perspective, DGPs transform inputs using a cascade of GPs such that the output of each layer of GPs forms the input to the GPs at the next layer, effectively implementing a deep probabilistic model for compositions of functions (Duvenaud et al., 2014; Neal, 1995).

Gaussian process composition was originally explored under the guise of hierarchical GP latent variable models (Lawrence and Moore, 2007) for the purpose of modelling dynamical systems with emphasis on human motion capture data, but DGPs were first rigorously formalised in the seminal work by Damianou and Lawrence (2013). In DGP models, the mapping between inputs and outputs is expressed as the composition of functions

$$\mathbf{f}(\mathbf{x}) = \left(\mathbf{f}^{(L)} \circ \ldots \circ \mathbf{f}^{(l)} \circ \ldots \circ \mathbf{f}^{(1)}\right)(\mathbf{x}), \tag{5.19}$$

where $\mathbf{f}^{(l)}$ denotes the latent variables at layer l, and each of the L layers is composed of a possibly transformed multivariate GP. In their presentation of DGPs, Bui et al. (2016) highlight two main arguments in favour of using DGPs instead of regular (henceforth shallow) GPs. The first point is based on the assertion that the effectiveness of GPs is tied to the suitability of the chosen kernel for capturing sensible correlations between datapoints. One could argue that given a sufficiently expressive kernel, there would be no need for the composition of functions characterising DGPs. Nevertheless, in the absence of highly-specialised domain experts, consistently constructing reliably good kernels is not a straightforward task. On the other hand, even though standard kernels may be used for the GPs at each layer, the structure of DGPs implicitly involves nonlinear transformations as well as expanding/reducing the dimensionality of inputs from one layer to the next, enabling the automatic construction of a more flexible model with greater representational capacity. Such composition also introduces non-stationarity in the model without explicitly using non-stationary kernels, which is the second most appealing quality of using DGPs.

However, although DGPs are attractive from a theoretical standpoint, inference is extremely challenging. Let us denote the set of latent variables at layer l by $\boldsymbol{F}^{(l)}$, and the conditional likelihood by $p\left(Y|F^{(L)}\right)$. Learning and making predictions with DGPs requires solving integrals that are generally intractable; for example, computing the marginal likelihood to optimise covariance parameters $\boldsymbol{\theta}^{(l)}$ at all layers entails solving

$$\begin{aligned} p(Y|X,\boldsymbol{\theta}) = \int & p\left(Y|F^{(L)}\right) p\left(F^{(L)}|F^{(L-1)},\boldsymbol{\theta}^{(L-1)}\right) \\ & \times \cdots \times p\left(F^{(1)}|X,\boldsymbol{\theta}^{(0)}\right) \mathrm{d}F^{(L)} \ldots \mathrm{d}F^{(1)}. \end{aligned} \tag{5.20}$$

This cannot be solved analytically and mandates the use of approximations in order to recover tractability. In this chapter, we introduce a novel approximation to DGPs that enables the application of DGPs to both regression and multi-class classification tasks having millions of observations, and consider set-ups with up to 30 hidden layers. This is well beyond the scale of datasets and architectures previously considered feasible for DGPs.

5.3.1 Preceding Work on Approximating DGPs

The first formulation of DGPs by Damianou and Lawrence (2013) relied on a variational inference scheme for propagating uncertainty across layers, along with inducing points at each layer in order to preserve tractability. The resulting model has the flavour of a mean-

field variational approximation since it assumes independence both between and across layers, which on the downside is likely to result in an oversimplification of the true DGP posterior. This approximation has a computational complexity of $\mathcal{O}\left(M^2LN\right)$. Given that the complexity only scales linearly with L, this encourages the increase of representational complexity to be achieved by way of adding more layers rather than using more inducing points at each layer. The storage requirements of the model, $\mathcal{O}\left(NL + M^2L\right)$, can be very restrictive however, and presents a major bottleneck to scalability. In fact, the evaluation of the original formulation of DGPs was limited to regression problems having few hundreds of observations at most.

An extension to the original DGP model put forward by Dai et al. (2016) borrows from the literature on autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) to develop a variational auto-encoded model whereby a recognition model is employed for constraining the variational posterior distribution of the latent variables. Although the model is applicable to both supervised and unsupervised learning problems, that work is primarily targeted towards investigating the latter by way of data imputation tasks, while only small regression datasets are considered once again.

On the other hand, the DGP approximation developed by Bui et al. (2016) is targeted towards supervised learning problems with particular emphasis on regression tasks. Contrary to the variational approximations detailed above, the proposed model relies on an expectation propagation scheme (EP; Minka, 2001) for estimating the marginal likelihood of a DGP. The required moment-matching is then carried out using nested assumed density filtering (Hernández-Lobato and Adams, 2015). This approach has similar time complexity to the original DGP framework, but lightens the memory requirement to $\mathcal{O}\left(M^2L\right)$. On a suite of regression problems, this approach outperforms both standard DGP approaches as well as a variation of Bayesian neural nets adapted with the reparameterisation trick (Kingma and Welling, 2014), hybrid Monte Carlo (Neal, 1992) and stochastic gradient Langevin dynamics (Welling and Teh, 2011). Even so, the experimental evaluation does not clearly compare the speed of the proposed method to competing techniques, while preliminary results on binary classification tasks indicate that the model does not perform as well on such problems. Nonetheless, given its superior predictive performance and amenability to mini-batch-based hyperparameter optimisation, we select this model as the primary DGP baseline against which to compare our proposal.

5.4 Practical Learning of Deep Gaussian Processes via Random Features

As highlighted in the previous section, existing inference approaches for DGP models have limited scalability and are notoriously cumbersome to construct. In this chapter, we develop a practical learning framework for training DGP models that significantly improves upon the state-of-the-art on those aspects. We show that random feature expansions for DGP models yield Bayesian neural networks (BNNs; Neal, 1995) with low-rank weight matrices, and the expansion of different covariance functions results in differing network activation functions, namely trigonometric for the RBF kernel and rectified linear unit (ReLU) functions for the arc-cosine covariance. In order to retain a probabilistic treatment of the model when scaling to large datasets, we employ stochastic variational inference techniques. In particular, we adapt the work on variational inference for neural networks and variational auto-encoders (Graves, 2011; Kingma and Welling, 2014) using mini-batch-based stochastic gradient optimisation, which can exploit GPU and distributed computing frameworks. In this respect, we can view the probabilistic treatment of DGPs approximated through random feature expansions as a means to specify sensible and interpretable priors for BNNs. Furthermore, unlike popular inducing point approximations for DGPs, the resulting learning framework does not involve any matrix decompositions in the size of the number of inducing points, but only matrix products.

5.4.1 Random Feature Expansions for Gaussian Processes

We start by describing how random feature expansions can be used to approximate the covariance of a shallow GP model. An in-depth introduction to kernel expansion via random features has already been provided in Section 2.3 of Chapter 2, with particular emphasis on the RBF kernel. Here we shall briefly recap this result, and additionally show how the arc-cosine covariance may also be expressed in a similar manner. We also give a brief overview of how recent advances have also made it possible to incorporate nonstationary covariances in this framework, although we do not delve into this aspect in our evaluation. For the sake of clarity, we initially present the covariances without any explicit scaling of the features or the covariance itself. After explaining the random feature expansion associated with each covariance, we then generalise these results in the context of DGPs to include scaling the covariances by a factor σ^2, as well as feature scaling for ARD.

RBF Covariance

Ignoring hyperparameters, recall that the RBF covariance function can be expressed as

$$k_{\text{RBF}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left[-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_j\right)^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)\right]. \tag{5.21}$$

Appealing to Bochner's theorem, any continuous shift-invariant normalised covariance function $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = k\left(\mathbf{x}_i - \mathbf{x}_j\right)$ is positive definite if and only if it can be rewritten as the Fourier transform of a non-negative measure $p(\omega)$ (Rahimi and Recht, 2008). Denoting the spectral frequencies by ω, while assigning $\imath = \sqrt{-1}$ and $\delta = \mathbf{x}_i - \mathbf{x}_j$, the covariance in Equation 5.21 can be expressed as

$$k_{\text{RBF}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \int p(\omega) \exp\left(\imath \delta^\top \omega\right) \mathrm{d}\omega, \tag{5.22}$$

with a corresponding non-negative measure $p(\omega) = \mathcal{N}(0, I)$. Because the covariance function and the non-negative measure are real, we can drop the unnecessary complex part of the argument, keeping $\cos\left(\delta^\top \omega\right) = \cos\left(\left(\mathbf{x}_i - \mathbf{x}_j\right)^\top \omega\right)$ that can in turn be rewritten as $\left[\cos\left(\mathbf{x}_i^\top \omega\right) \cos\left(\mathbf{x}_j^\top \omega\right) + \sin\left(\mathbf{x}_i^\top \omega\right) \sin\left(\mathbf{x}_j^\top \omega\right)\right]$.

The importance of the expansion above is that it allows us to interpret the covariance function as an expectation that can be estimated using MC sampling. Defining $\mathbf{z}(\mathbf{x}|\omega) = \left[\cos\left(\mathbf{x}^\top \omega\right), \sin\left(\mathbf{x}^\top \omega\right)\right]^\top$, the covariance function can therefore be unbiasedly approximated as

$$k_{\text{RBF}}\left(\mathbf{x}_i, \mathbf{x}_j\right) \approx \frac{1}{N_{\text{RF}}} \sum_{r=1}^{N_{\text{RF}}} \mathbf{z}\left(\mathbf{x}_i|\widetilde{\omega}_r\right)^\top \mathbf{z}\left(\mathbf{x}_j|\widetilde{\omega}_r\right), \tag{5.23}$$

with N_{RF} random samples $\widetilde{\omega}$ sampled from $p(\omega)$. As discussed in Chapter 2, the sparse spectrum GP approximation investigated by Lázaro-Gredilla et al. (2010) is based on this spectral representation of the RBF kernel.

Arc-cosine Covariance

We now consider the arc-cosine covariance function introduced in Section 5.2.2 of this chapter, which for order d and $l = 1$ is given by

$$k_{\text{ARC}}^d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{1}{\pi}\left(\|\mathbf{x}_i\| \|\mathbf{x}_j\|\right)^d J_d\left(\cos^{-1}\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}\right)\right), \tag{5.24}$$

where we have defined

$$J_d(\alpha) = (-1)^d (\sin\alpha)^{2d+1} \left(\frac{1}{\sin\alpha}\frac{\partial}{\partial\alpha}\right)^d \left(\frac{\pi-\alpha}{\sin\alpha}\right). \tag{5.25}$$

Let $\mathcal{H}(\cdot)$ be the Heaviside function; following Cho and Saul (2009), an integral representation of this covariance is given by

$$\begin{aligned} k^d_{\text{ARC}}(\mathbf{x}_i, \mathbf{x}_j) = 2 \int \mathcal{H}\left(\omega^\top \mathbf{x}_i\right) \left(\omega^\top \mathbf{x}_i\right)^d \mathcal{H}\left(\omega^\top \mathbf{x}_j\right) \left(\omega^\top \mathbf{x}_j\right)^d \\ \times \mathcal{N}(\omega|0, I)\, \mathrm{d}\omega. \end{aligned} \tag{5.26}$$

This integral formulation immediately suggests a random feature approximation for the arc-cosine covariance in Equation 5.24, noting that it can be seen as an expectation of the product of the same function applied to the inputs to the covariance. As before, this results in an approximate explicit representation of the mapping inducing the covariance function. Interestingly, for the arc-cosine covariance of order $d = 1$, this yields an approximation based on the popular rectified linear unit (ReLU) functions, which are widely-used for constructing neural networks. We note that for the arc-cosine covariance with degree $d = 0$, the resulting Heaviside activations are unsuitable for our inference scheme given that they systematically yield zero gradients.

Nonstationary Covariance Functions

In our presentation of GPs thus far, we have predominantly assumed the chosen kernel to be stationary, i.e. the covariance function relies on the distance between inputs rather than their actual location. However, in applications such as geostatistics and time series forecasting, the absolute location of the inputs to the covariance should also be incorporated in the kernel function. Whereas Bochner's theorem (see Section 2.3.1 in Chapter 2) applies to stationary kernels, there has been an ongoing effort to generalise these results to cover nonstationary behaviour. Preliminary work by Kom Samo and Roberts (2015) explored the theoretical implications this entails, and presented a more general flexible representation of nonstationary kernels using random features. Subsequent work by Remes et al. (2017) tailored these ideas to the GP regression setting, with particular emphasis on a nonstationary variation of the spectral mixture kernel originally proposed by Wilson and Adams (2013). Similar constructions with emphasis on Gaussian process regression have also recently been investigated by Ton et al. (2018). We do not follow up on this particular family

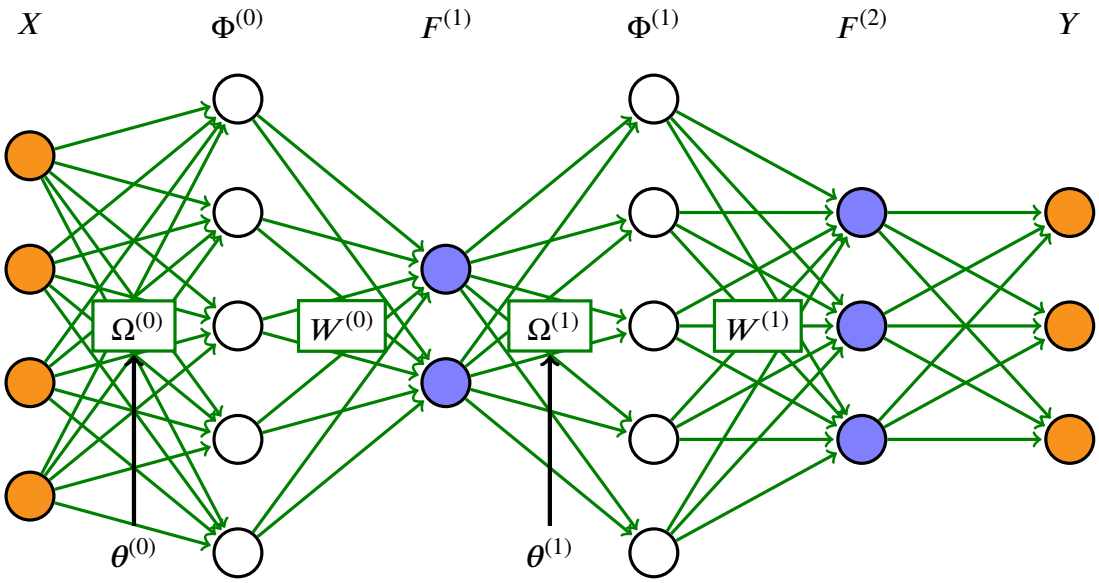


Fig. 5.1 The proposed DGP approximation. At each hidden layer GPs are replaced by their two-part weight-space approximation. The random features $\Phi^{(l)}$ are obtained using a weight matrix $\Omega^{(l)}$. This is followed by a linear transformation parameterised by weights $W^{(l)}$. The prior over $\Omega^{(l)}$ is determined by the covariance parameters $\theta^{(l)}$ of the original GPs.

of kernels in this thesis; however, this should be an interesting avenue for future work to develop further using the foundations established here.

5.4.2 Extension to Deep Gaussian Processes

We now present a novel approximate formulation of DGPs which, as we illustrate in the subsequent experimental evaluation, leads to a practical learning algorithm for these deep probabilistic nonparametric models. In particular, we propose to employ a random feature expansion at each layer, and in doing so we obtain an approximation to the original DGP model as a deep neural network. An overview of this architecture is illustrated in Figure 5.1.

Let us assume that the GPs have zero mean, and define $F^{(0)} = X$. We also assume that the GP covariances at each layer are parametrised through a set of parameters $\theta^{(l)}$. The parameter set $\theta^{(l)}$ comprises the layer-wise GP marginal variances $\left(\sigma^2\right)^{(l)}$ and lengthscale parameters

$$\Lambda^{(l)} = \operatorname{diag}\left[\left(l_1^2\right)^{(l)}, \ldots, \left(l^2_{D_F^{(l)}}\right)^{(l)}\right]. \tag{5.27}$$

Considering a DGP with RBF covariances, taking a weight-space view (see Chapter 2, Section 2.3) of the GPs at each layer, and extending the results of the previous section, we have that

$$\Phi_{\text{RBF}}^{(l)} = \sqrt{\frac{\left(\sigma^2\right)^{(l)}}{N_{\text{RF}}^{(l)}}} \left[\cos\left(F^{(l)}\Omega^{(l)}\right), \sin\left(F^{(l)}\Omega^{(l)}\right)\right], \tag{5.28}$$

and $F^{(l+1)} = \Phi_{\text{RBF}}^{(l)} W^{(l)}$. At each layer, the priors over the random weights are $p\left(\Omega_{\cdot j}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$ and $\left(W^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, I_{D_F^{(l+1)}}\right)$. Each matrix $\Omega^{(l)}$ has dimensionality $D_F^{(l)} \times N_{\text{RF}}^{(l)}$. On the other hand, the weight matrices $W^{(l)}$ have dimensions $2N_{\text{RF}}^{(l)} \times D_F^{(l+1)}$ for weighing the sine and cosine random features, with the constraint that $D_F^{(L)} = D_{\text{out}}$.

Similarly, considering a DGP with arc-cosine covariances of order $d = 1$, the application of the random feature approximation leads to an architecture with ReLU activations,

$$\Phi_{\text{ARC}}^{(l)} = \sqrt{\frac{2\left(\sigma^2\right)^{(l)}}{N_{\text{RF}}^{(l)}}} \max\left(0, F^{(l)}\Omega^{(l)}\right), \tag{5.29}$$

with $\Omega_{\cdot j}^{(l)} \sim \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$, which are cheaper to evaluate and differentiate than the trigonometric functions required for RBF. As in that case, we can also allow the covariance and the features to be scaled by $\left(\sigma^2\right)^{(l)}$ and $\Lambda^{(l)}$ respectively. The dimensions of the weight matrices $\Omega^{(l)}$ are the same as in the RBF case, but the dimensions of the $W^{(l)}$ matrices are reduced to $N_{\text{RF}}^{(l)} \times D_F^{(l+1)}$.

5.4.3 Network Architecture with Low-rank Weights

Our formulation of an approximate DGP using random feature expansions reveals a close connection to deep neural networks (DNNs). In our formulation, the design matrices at each layer are $\Phi^{(l+1)} = \gamma\left(\Phi^{(l)} W^{(l)} \Omega^{(l+1)}\right)$, where $\gamma(\cdot)$ denotes the element-wise application of covariance-dependent functions, i.e. sine and cosine for RBF, and ReLU for the first order arc-cosine kernel. Instead, for a regular DNN, the design matrices are computed as $\Phi^{(l+1)} = g\left(\Phi^{(l)}\Omega^{(l)}\right)$, where $g(\cdot)$ is a so-called activation function. From a probabilistic standpoint, we can thus interpret our approximate DGP model as a DNN with specific Gaussian priors over the $\Omega^{(l)}$ weights controlled by the covariance parameters $\theta^{(l)}$, and standard Gaussian priors over the $W^{(l)}$ weights. Covariance parameters act as hyper-priors over the weights $\Omega^{(l)}$, and the objective is to optimise these during training.

Another observation about the resulting DGP approximation is that, for a given layer l, the transformations given by $W^{(l)}$ and $\Omega^{(l+1)}$ are both linear. If we collapsed the two transformations into a single one, by introducing weights $\Xi^{(l)} = W^{(l)}\Omega^{(l+1)}$, we would have to learn $\mathcal{O}\left(N_{\text{RF}}^{(l)} \times N_{\text{RF}}^{(l+1)}\right)$ weights at each layer, which is considerably more expensive than learning the two separate sets of weights. As a result, we can view the proposed approximate DGP model as a means of imposing low-rank structure on the weights of the architecture, which is a form of regularisation that is frequently cited in the literature of DNNs (Denil et al., 2013; Sainath et al., 2013). This low-rank structure of DGPs was also noted by Duvenaud (2014) and Damianou (2015).

The connection between DGPs and DNNs has been pointed out in several papers, such as Neal (1995) and Duvenaud et al. (2014), where the pathologies that may arise in such deep models are investigated. Dropout is another technique intended to speed up training and improve regularisation in neural networks that has recently been linked to variational inference (Gal and Ghahramani, 2016) by way of MC sampling. The additional links introduced in that work in relation to DGPs shall be highlighted later in this chapter.

5.4.4 Stochastic Variational Inference for Deep Gaussian Processes

In order to keep the notation uncluttered, let $\boldsymbol{\Theta}$ be the collection of all covariance parameters $\boldsymbol{\theta}^{(l)}$ at all layers. We shall first consider the setting of a DGP with fixed spectral frequencies $\Omega^{(l)}$ collected into $\boldsymbol{\Omega}$, and let $\mathbf{W}$ be the collection of the weight matrices $W^{(l)}$ at all layers. For $\mathbf{W}$, we have a product of standard normal priors stemming from the approximation of the GPs at each layer, $p(\mathbf{W}) = \prod_{l=1}^{L} p\left(W^{(l)}\right)$, and we propose to treat W using variational inference following Graves (2011) and Kingma and Welling (2014), while optimising all covariance parameters $\boldsymbol{\Theta}$. For ease of exposition, we initially consider $\boldsymbol{\Omega}$ to be fixed here, but will discuss alternative ways to treat this set of parameters in the next section.

The marginal likelihood $p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta})$ involves intractable integrals, but we can obtain a tractable lower bound using variational inference and applying Jenson's inequality. Defining $\mathcal{E} = \mathbb{E}_{q(\mathbf{W})} \log\left[p(Y|X, \mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\Theta})\right]$, we obtain

$$
\begin{aligned}
\log[p(Y|X,\boldsymbol{\Omega},\boldsymbol{\Theta})] &= \log\left[\int p(Y|X,\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Theta})p(\mathbf{W})\mathrm{d}\mathbf{W}\right] \\
&= \log\left[\int \frac{p(Y|X,\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Theta})p(\mathbf{W})}{q(\mathbf{W})}q(\mathbf{W})\mathrm{d}\mathbf{W}\right] \\
&= \log\left[\mathbb{E}_{q(\mathbf{W})}\frac{p(Y|X,\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Theta})p(\mathbf{W})}{q(\mathbf{W})}\right] \\
&\geq \mathbb{E}_{q(\mathbf{W})}\left(\log\left[\frac{p(Y|X,\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Theta})p(\mathbf{W})}{q(\mathbf{W})}\right]\right) \\
&= \mathbb{E}_{q(\mathbf{W})}\left(\log[p(Y|X,\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Theta})]\right) + \mathbb{E}_{q(\mathbf{W})}\left(\log\left[\frac{p(\mathbf{W})}{q(\mathbf{W})}\right]\right) \\
&= \mathcal{E} - D_{\mathrm{KL}}\left[q(\mathbf{W})||p(\mathbf{W})\right], \qquad (5.30)
\end{aligned}
$$

where $q(\mathbf{W})$ acts as a variational approximation to the posterior over all the weights $p(\mathbf{W}|Y,X,\boldsymbol{\Omega},\boldsymbol{\Theta})$.

We are interested in optimising $q(\mathbf{W})$, i.e. finding an optimal approximate distribution over the parameters according to the bound given in Equation 5.30. As with previous mentions of the evidence lower bound on the marginal likelihood in this thesis, the first term can be interpreted as a model fit term whereas the second can be seen as regularisation. In the case of a Gaussian distribution $q(\mathbf{W})$, it is possible to compute the D_{KL} term analytically. More specifically, given two Gaussian distributions $p_1(x) = \mathcal{N}\left(\mu_1, \sigma_1^2\right)$ and $p_2(x) = \mathcal{N}\left(\mu_2, \sigma_2^2\right)$, the KL divergence between them is given by

$$
D_{\mathrm{KL}}\left[p_1(x)\,\|p_2(x)\right] = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1-\mu_2)^2}{\sigma_2^2}\right]. \qquad (5.31)
$$

To guarantee tractability, we shall consider a mean-field variational approximation that factorises across both layers and weights,

$$
q(\mathbf{W}) = \prod_{ijl} q\left(W_{ij}^{(l)}\right) = \prod_{ijl} \mathcal{N}\left(m_{ij}^{(l)}, \left(s^2\right)_{ij}^{(l)}\right). \qquad (5.32)
$$

The variational parameters are the mean and variance of each of the approximating factors, i.e. $m_{ij}^{(l)}$ and $\left(s^2\right)_{ij}^{(l)}$, and we aim to optimise the lower bound with respect to these as well as all covariance parameters grouped in $\boldsymbol{\Theta}$.

However, the $\mathcal{E}$ term will still have to be approximated regardless. In the case of a likelihood that factorises across observations, an interesting feature of this expression of the lower bound is that it is amenable to fast stochastic optimisation. In particular, we

derive a doubly-stochastic approximation of the expectation in the lower bound as follows. First, $\mathcal{E}$ can be rewritten as a sum over the input points, which allows us to estimate it in an unbiased fashion using mini-batches, each time selecting M points indexed by $\mathcal{I}_{\text{MB}}$,

$$\mathcal{E} \approx \frac{N}{M} \sum_{i \in \mathcal{I}_{\text{MB}}} \mathbb{E}_{q(\mathbf{W})} \left(\log \left[p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{W}, \mathbf{\Omega}, \mathbf{\Theta}) \right] \right). \tag{5.33}$$

Subsequently, each of the elements of the sum can be estimated using $N_{\mathbf{s}}$ MC samples, yielding

$$\mathcal{E} \approx \frac{N}{M} \sum_{i \in \mathcal{I}_{\text{MB}}} \frac{1}{N_{\mathbf{s}}} \sum_{s=1}^{N_{\mathbf{s}}} \log \left[p(\mathbf{y}_i | \mathbf{x}_i, \widetilde{\mathbf{W}}_s, \mathbf{\Omega}, \mathbf{\Theta}) \right], \tag{5.34}$$

with $\widetilde{\mathbf{W}}_s \sim q(\mathbf{W})$. In order to facilitate the optimisation, we once again invoke the reparameterisation trick (Kingma and Welling, 2014) for rewriting the weights as follows,

$$\left(\widetilde{W}_s^{(l)} \right)_{ij} = m_{ij}^{(l)} + \varepsilon_{sij}^{(l)} s_{ij}^{(l)}. \tag{5.35}$$

By differentiating the lower bound with respect to $\mathbf{\Theta}$ and the mean and variance of the approximate posterior over $\mathbf{W}$, we obtain an unbiased estimate of the gradient for the lower bound. The reparameterisation trick ensures that the randomness in the computation of the expectation is fixed when applying stochastic gradient ascent moves to the parameters of $q(\mathbf{W})$ and $\mathbf{\Theta}$ (Kingma and Welling, 2014). Automatic differentiation tools then enable us to compute stochastic gradients automatically, which is why we opted to implement our model in TensorFlow (Abadi et al., 2015).

5.4.5 Treatment of Spectral Frequencies

So far, we have assumed the spectral frequencies $\mathbf{\Omega}$ to be sampled from the prior and fixed throughout, whereby we employ the reparameterisation trick to obtain $\Omega_{ij}^{(l)} = \mu_{ij}^{(l)} + \varepsilon_{sij}^{(l)} \left(\beta^2\right)_{ij}^{(l)}$, with $\mu_{ij}^{(l)}$ and $\left(\beta^2\right)_{ij}^{(l)}$ determined by the prior $p\left(\Omega_{\cdot j}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, \left(\Lambda^{(l)}\right)^{-1}\right)$. We then draw the $N_{\mathbf{s}}$ $\varepsilon_{sij}^{(l)}$ elements and fix them from the outset, such that the covariance parameters $\mathbf{\Theta}$ can be optimised along with $q(\mathbf{W})$. We refer to this variant as PRIOR-FIXED.

Inspired by previous work on random feature expansions for GPs, e.g. Lázaro-Gredilla et al. (2010) and Gal and Turner (2015), we can think of alternative ways to treat these parameters. In particular, we study a variational treatment for $\mathbf{\Omega}$; defining $\mathbf{\Psi} = [\mathbf{W}, \mathbf{\Omega}]$,

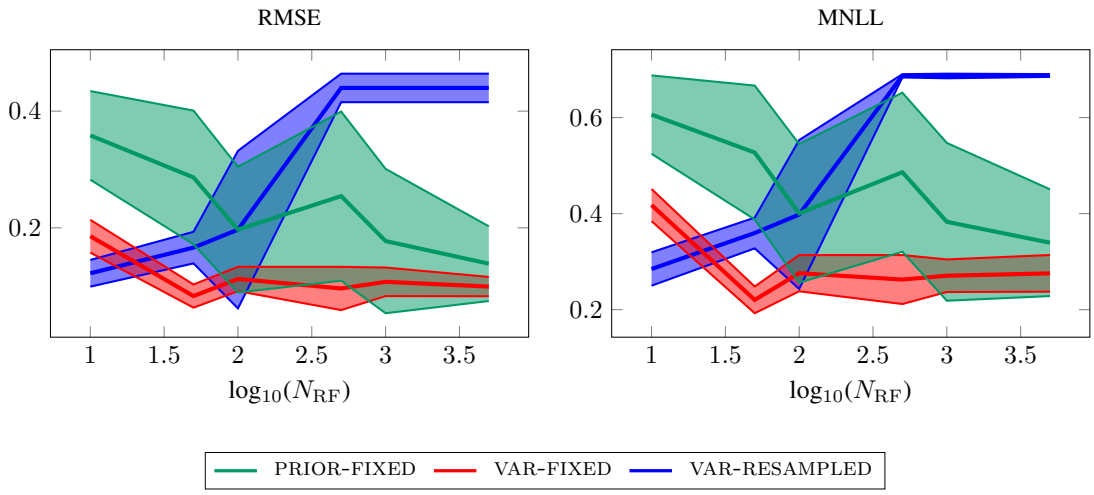


Fig. 5.2 Performance of different strategies for dealing with $\mathbf{\Omega}$ as a function of the number of random features. These can be fixed (PRIOR-FIXED), or treated variationally (with fixed randomness VAR-FIXED or resampled at each iteration VAR-RESAMPLED).

we can derive an alternate lower bound to the one presented in Equation 5.30 such that

$$\log\left[p\left(Y|X,\mathbf{\Psi},\mathbf{\Theta}\right)\right] \geq \mathbb{E}_{q(\mathbf{\Psi})}\left(\log\left[p\left(Y|X,\mathbf{\Psi},\mathbf{\Theta}\right)\right]\right) - D_{\mathrm{KL}}\left[q(\mathbf{\Psi})||p(\mathbf{\Psi}|\mathbf{\Theta})\right]. \tag{5.36}$$

Once again assuming a factorised prior over all frequencies results in

$$p\left(\mathbf{\Psi}|\theta\right) = \prod_{l=0}^{L-1} p\left(\Omega^{(l)}|\theta^{(l)}\right) p\left(W^{(l)}\right) \approx \prod_{ijl} q\left(\Omega_{ij}^{(l)}\right) \prod_{ijl} q\left(W_{ij}^{(l)}\right). \tag{5.37}$$

When being variational about $\mathbf{\Omega}$, we introduce an approximate posterior $q(\mathbf{\Omega})$ which also has a factorised form. We use the reparameterisation trick once again, but $\mathbf{\Omega}$ are now sampled from the posterior, which in general has different mean and variances to the prior. We report two variations of this treatment, namely VAR-FIXED and VAR-RESAMPLED. In VAR-FIXED, we fix $\epsilon_{sij}^{(l)}$ in computing $\mathbf{\Omega}$ when training the model, whereas in VAR-RESAMPLED we resample these at each iteration. We note that one can also be variational about $\mathbf{\Theta}$, but do not investigate this option here.

In Figure 5.2, we illustrate the differences between the strategies discussed in this section, i.e. fixing the spectral frequencies $\mathbf{\Omega}$ or treating them variationally. We report the accuracy of a one-layer DGP with RBF covariances with respect to the number of random features on one of the datasets that we consider in the experiments section (EEG). For PRIOR-FIXED, more random features result in a better approximation of the GP pri-

ors at each layer, and this results in better generalisation. When we resample $\mathbf{\Omega}$ from the approximate posterior (VAR-RESAMPLED), we notice that the model quickly struggles with the optimisation as the number of random features increases. We attribute this to the fact that the factorised form of the posterior over $\mathbf{\Omega}$ and $\mathbf{W}$ is unable to capture posterior correlations between the coefficients for the random features and the weights of the corresponding linearised model. Being deterministic about the way spectral frequencies are computed (VAR-FIXED) offers the best performance among the three learning strategies, and this is what we employ throughout the remainder of this chapter.

5.4.6 Computational Complexity

When estimating the lower bound, the most computationally expensive operations are matrix multiplications performed at each layer, specifically $\boldsymbol{F}^{(l)}\boldsymbol{\Omega}^{(l)}$ and $\boldsymbol{\Phi}^{(l)}\boldsymbol{W}^{(l)}$. Recalling that this matrix product is done for samples from the posterior over $\mathbf{W}$ (and also $\mathbf{\Omega}$ when treated variationally) and given the mini-batch formulation, the former costs $\mathcal{O}\left(N_{\mathbf{s}} M D_F^{(l)} N_{\mathrm{RF}}^{(l)}\right)$, while the latter costs $\mathcal{O}\left(N_{\mathbf{s}} M N_{\mathrm{RF}}^{(l)} D_F^{(l+1)}\right)$.

Due to the interpretation using random feature expansions combined with stochastic variational inference, the resulting algorithm does not involve any Cholesky decompositions. This is in sharp contrast with stochastic variational inference using inducing-point approximations (see e.g. Bui et al., 2016; Dai et al., 2016), where such operations could significantly limit the number of inducing points that can be employed. The computational speed-up resulting from bypassing Cholesky decompositions will be further emphasised in the next section.

5.5 Experimental Evaluation

We evaluate our model by comparing it against relevant alternatives for both regression and classification tasks, and assess its performance when applied to large-scale datasets. We also investigate the extent to which such deep compositions continue to yield good performance when the number of hidden layers is significantly increased.

5.5.1 Model Comparison

We primarily compare our model to state-of-the-art DGPs constructed using an expectation propagation framework (DGP-EP; Bui et al., 2016). The comparison originally included results for the variational auto-encoded DGP by Dai et al. (2016), but the results

obtained using the available code were not competitive with DGP-EP and we thus decided to exclude them from the figures. We also omitted DGP training using sequential inference (Wang et al., 2016) since the performance reported in the paper is inferior to more recent approaches. In order to present our results in a wider context, and demonstrate that DGPs indeed lead to better quantification of uncertainty, we also compare against DNNs trained using dropout. Finally, to substantiate the benefits of using deep models, we set the shallow sparse variational GP (Hensman et al., 2015a) implemented in GPflow (Matthews et al., 2017) as the baseline against which all other models are compared.

We use the same experimental set-up for both regression and classification tasks using datasets obtained from the UCI dataset repository (Asuncion and Newman, 2007) for models having one and two hidden layers. The specific configurations for each model are detailed below:

- **DGP-RBF**, **DGP-ARC** : In the proposed DGP with an RBF kernel, we use 100 random features at every hidden layer to construct a multivariate GP with $D_F^{(l)} = 3$, and set the batch size to $M = 200$. We initially only use a single MC sample to kick-start the optimisation procedure, but this is then increased to 100 samples halfway through the allocated optimisation time in order to ensure better convergence and stability. We employ the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.01, and in order to stabilise the optimisation procedure, we fix the parameters $\boldsymbol{\Theta}$ for 12,000 iterations before jointly optimising all parameters. As discussed in Section 5.4.5, $\boldsymbol{\Omega}$ are optimised variationally with fixed randomness unless stated otherwise. The same set-up is used for DGP-ARC, the variation of our model implementing the arc-cosine kernel;

- **DGP-EP** [1]: For this technique, we use the same architecture and optimiser as for DGP-RBF and DGP-ARC, a batch size of 200, and 100 inducing points at each layer. For the classification case, we use 100 samples for approximating the Softmax likelihood;

- **DNN** : We construct a DNN configured with a dropout rate of 0.5 at each hidden layer so as to provide regularisation during training. In order to preserve a degree of fairness, we set the number of weights to be optimised so as to match those in the DGP-RBF and DGP-ARC models when the random features are assumed to be fixed.

[1] Code obtained from: https://github.com/thangbui/deepGP_approxEP

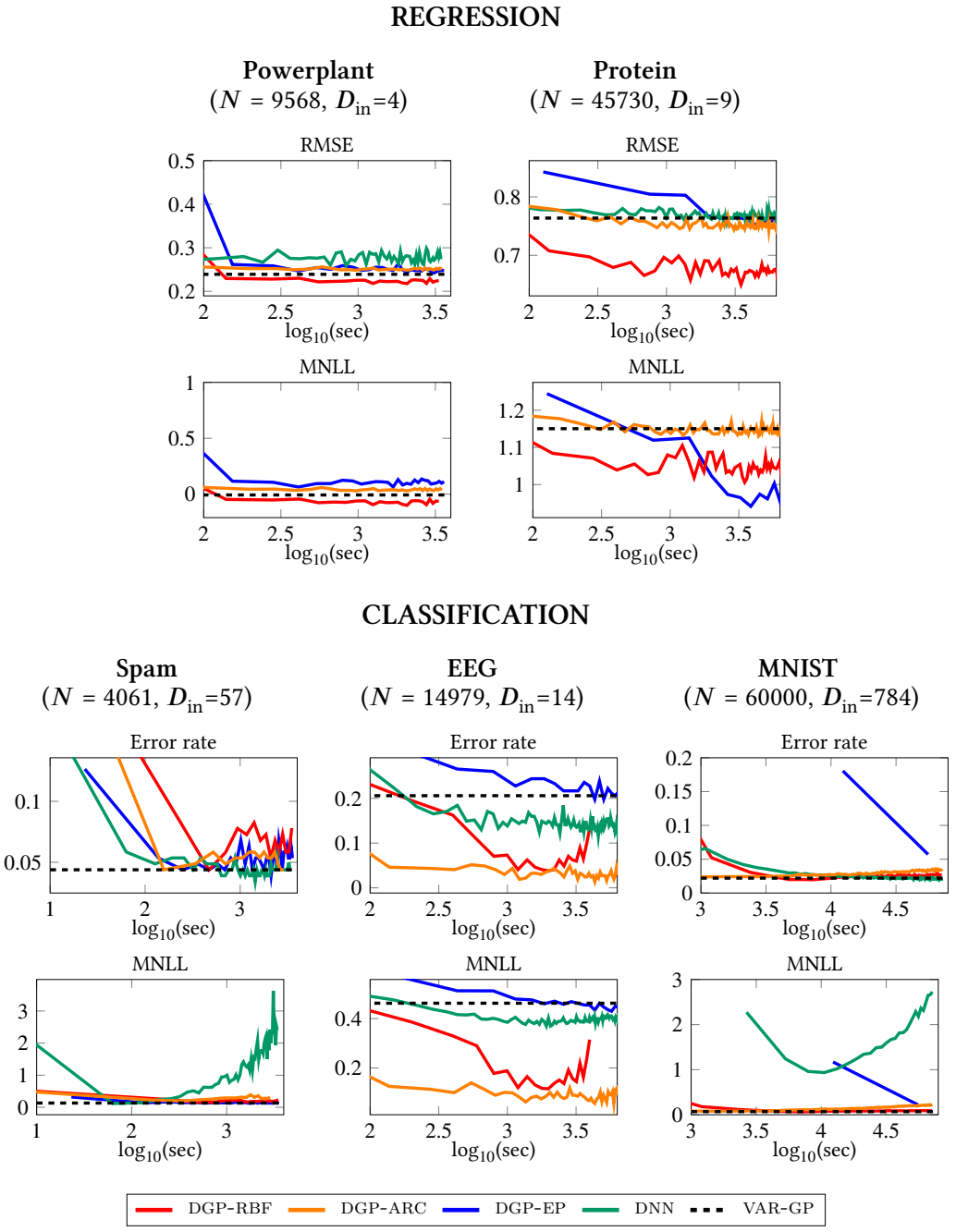


Fig. 5.3 Progression of error rate (RMSE in the regression case) and MNLL over time for competing models. Results are shown for configurations having 1 hidden layer.

We assess the performance of each model using the error rate (or RMSE in the regression case) and mean negative log likelihood (MNLL) on withheld test data. The experiments were launched on single nodes of a cluster of Intel Xeon E5-2630 CPUs having 32 cores and 128GB RAM, and the results are averaged over 3 folds for every dataset.

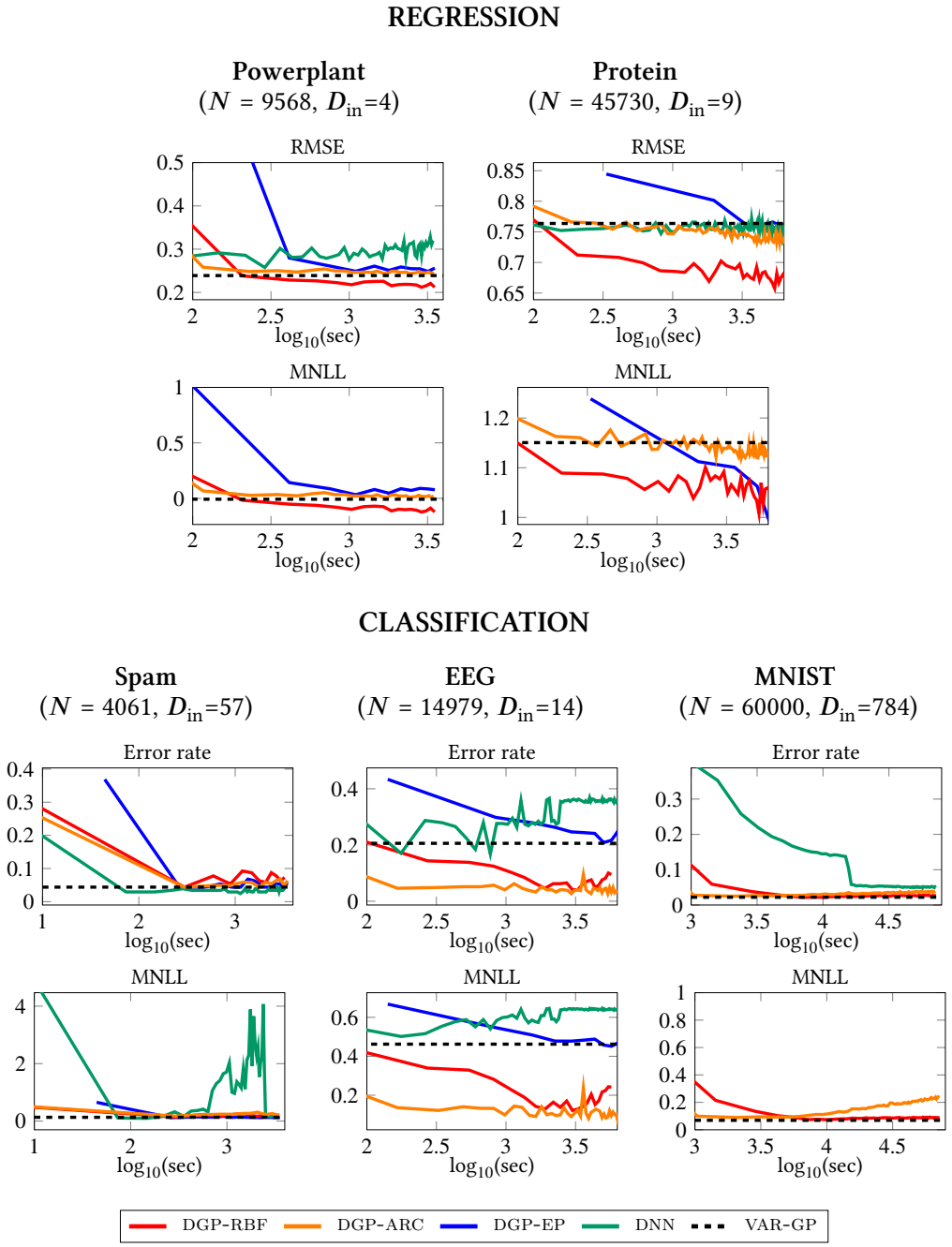


Fig. 5.4 Progression of error rate (RMSE in the regression case) and MNLL over time for competing models. Results are shown for configurations having 2 hidden layers.

Figure 5.3 shows that DGP-RBF and DGP-ARC consistently outperform competing techniques both in terms of convergence speed and predictive accuracy. This is particularly significant for larger datasets where other techniques take considerably longer to converge to a reasonable error rate, although DGP-EP converges to superior MNLL for the Protein dataset. The results are also competitive (and often superior) to those obtained by the variational GP (VAR-GP) presented in Hensman et al. (2015a). It is striking to see how inferior the uncertainty quantification obtained for the DNN (which is inherently limited to the classification case, so no MNLL reported on regression datasets) is in comparison to DGPs, despite the error rate being comparable. This further reinforces the underlying theme of the thesis whereby we champion the use of Bayesian machine learning models over other approaches.

By virtue of its higher dimensionality, larger configurations were used for MNIST. For DGP-RBF and DGP-ARC, we use 500 random features, 50 GPs in the hidden layers, batch size of 1000, Adam with a 0.001 learning rate, and fix $\boldsymbol{\Theta}$ for $12,000$ iterations. Similarly for DGP-EP, we use 500 inducing points, with the only difference being a slightly smaller batch size to cater for issues with memory requirements. Following Simard et al. (2003), we employ 800 hidden units at each layer of the DNN. The DGP-RBF model peaks at 98.04% and 97.93% accuracy for one and two hidden layers respectively. It was observed that the model performance degrades noticeably when more than 2 hidden layers are used (without feeding forward the inputs). This is in line with what is reported in the literature on DNNs (Duvenaud et al., 2014; Neal, 1995). By simply re-introducing the original inputs in the hidden layer, the accuracy improves to 98.2% for the one hidden layer case.

Recent experiments on MNIST using a variational GP with MCMC report overall accuracy of 98.04% (Hensman et al., 2015a), while the AutoGP architecture presented earlier in this chapter has been shown to give 98.45% accuracy. Using a finer-tuned configuration, DNNs can reportedly obtain 98.4% accuracy (Simard et al., 2003), whereas 98.6% has been reported for SVMs (Schölkopf, 1997). In view of this wider scope of inference techniques, it can be confirmed that the results obtained using the proposed architecture are comparable to the state-of-the-art, even if further extensions may be required for consistently obtaining a proper edge. Note that this comparison focuses on approaches that do not carry out any preprocessing on the data, and excludes convolutional neural nets.

5.5.2 Large-scale Datasets

One of the defining characteristics of this DGP model is the ability to scale up to large datasets without compromising on performance and accuracy in quantifying uncertainty.

Table 5.2 Performance of our DGP proposal on large-scale datasets.

Dataset	Accuracy		MNLL	
	RBF	ARC	RBF	ARC
MNIST-8M	99.14%	99.04%	0.0454	0.0465
Airline	78.55%	72.76%	0.4583	0.5335

As a demonstrative example, we evaluate our model on two large-scale problems beyond the scale to which GPs and especially DGPs are typically applied.

We first consider MNIST-8M, which artificially extends the original MNIST dataset to over 8 million observations. We trained this model using the same configuration described for standard MNIST, and obtained 99.14% accuracy on the test set using one hidden layer. Given the size of this dataset, there are only few reported results for other GP models; as noted in Section 5.2.4, 99.11% accuracy can be obtained with the AutoGP framework, which is comparable to the result obtained by this DGP model.

Meanwhile, the Airline dataset contains flight information for over 5 million flights in the US between January and April 2008. Following the procedure described in Hensman et al. (2013) and Wilson et al. (2016b), we use this 8-dimensional dataset for classification, where the task is to determine whether a flight has been delayed or not. We generate the test set using the scripts provided in Wilson et al. (2016b), where 100,000 data-points are held out for testing. Our DGP models are constructed using 100 random features at each layer, and set the dimensionality at each layer to $D_F^{(l)} = 3$. As shown in Table 5.2, our model works significantly better when the RBF kernel is employed. In addition, the results are also directly comparable to those obtained by Wilson et al. (2016b), which reports accuracy and MNLL of 78.1% and 0.457, respectively. These results give further credence to the tractability, scalability, and robustness of the proposed model.

5.5.3 Model Depth

In this final part of the experiments, we assess the scalability of our model with respect to additional hidden layers in the constructed model. In particular, we reconsider the Airline dataset introduced in the previous section and evaluate the performance of DGP-RBF models constructed using up to 30 layers. In order to cater for the increased depth in the

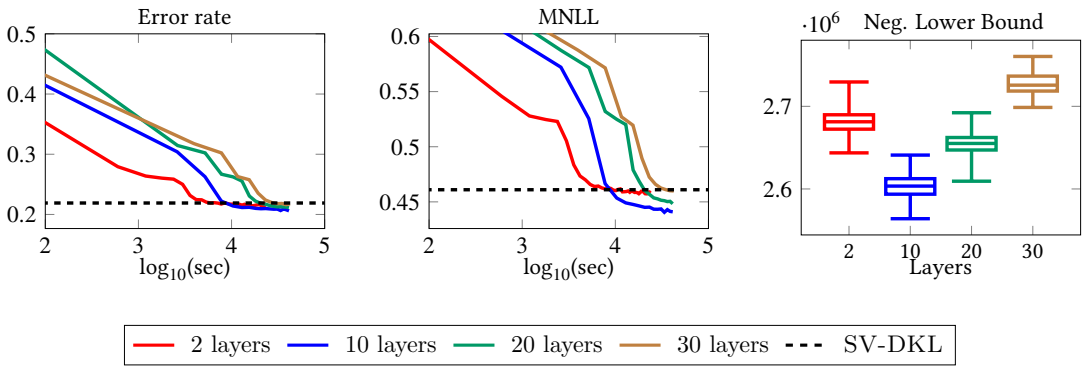


Fig. 5.5 Left and centre - Performance of our model on the Airline dataset as a function of time for different depths. Deeper models (10, 20 layers) can outperform shallow architectures after convergence. The baseline (SV-DKL) is taken from Wilson et al. (2016b). Right - The box plot of the negative evidence lower bound (estimated over 100 mini-batches of size 50,000) confirms this is a suitable objective for model selection.

model, we feed-forward the original input to each hidden layer, as suggested in Duvenaud et al. (2014) for avoiding pathologies.

Figure 5.5 reports the progression of error rate and MNLL over time for models constructed with an increasing number of hidden layers, using the results listed in Wilson et al. (2016b) as a baseline (reportedly obtained in about three hours). As expected, our DGP model takes longer to train as the number of layers increases. However, the model converges to an optimal state for every setting in less than a couple of hours, with an improvement being noted in the case of 10 and 20 layers over the shallower 2-layer model. The box-plot within the same figure confirms that the negative evidence lower bound evaluated on the training data is a suitable objective for carrying out model selection.

5.5.4 Distributed Implementation

This model is also easily amenable to a distributed implementation using asynchronous distributed stochastic gradient descent (Chilimbi et al., 2014). Our setting, based on the parameter server framework (Li et al., 2014) implemented in TensorFlow, includes one or more parameter servers and a number of workers. The latter proceed asynchronously using randomly selected batches of data; they fetch fresh model parameters from the parameter server, compute the gradients of the lower bound with respect to these parameters, and push those gradients back to the parameter server, which in turn updates the global representation of the model accordingly. Given that workers compute gradients and send updates to the parameter server asynchronously, the discrepancy between the model used

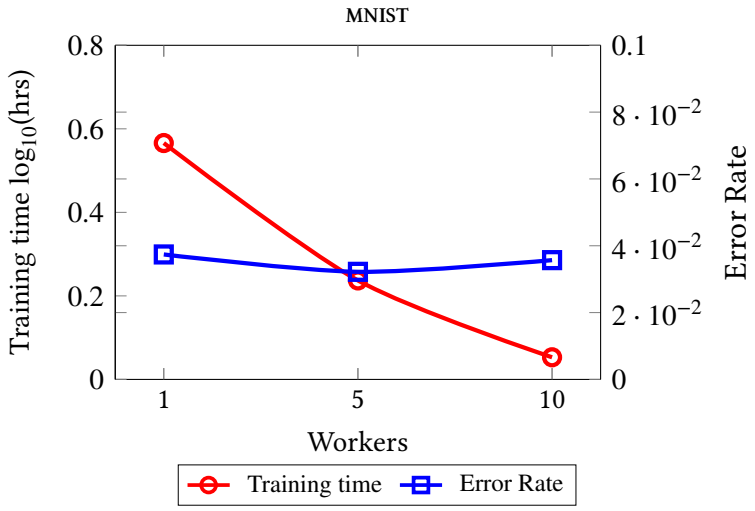


Fig. 5.6 Comparison of training time and error rate for asynchronous DGP-RBF with 2 parameter servers, and alternative settings having 1, 5 and 10 workers respectively.

to compute gradients and the local model can degrade training quality. This is exacerbated by having a large number of asynchronous workers, as noted in Chen et al. (2016).

We carry out a preliminary investigation into the viability of this training approach. In particular, we focus our evaluation on the MNIST dataset, and study how training time and error rate evolve in relation to the number of workers introduced in our set-up. The parameters for the model are identical to those reported for the previous experiments, except that we fix the number of MC samples to one throughout. We report the results in Figure 5.6. As expected, the training time decreases in proportion to the number of workers, albeit sub-linearly, but the error rate remains largely unchanged when varying the number of workers. When using a single parameter server, we noticed that increasing the number of workers led to lower gains in execution speed, and the behaviour of the error rate over iterations was more erratic than in the case of two parameter servers. We attribute these effects to the greater overheads on the communication cost when using a single parameter server, as well as the more involved coordination of multiple workers. The work in Chen et al. (2016) corroborates our findings, and motivates further work in the direction of alleviating this issue.

5.6 Impact and Extensions

When they were first published, both the AutoGP framework (Krauth et al., 2017) and the proposed DGP approximation (Cutajar et al., 2017) were key to showcasing how GPs and

DGPs can be adapted to the large-scale and very complex tasks to which machine learning is now being applied. Whereas GPs were once deemed unfit to compete with state-of-the-art deep learning techniques, these works contributed towards a broader shift in perspective regarding the capabilities and limitations of GPs. In fact, the results obtained by AutoGP for image classification datasets are often included as benchmarks against which more specialised models such as convolutional GPs (van der Wilk et al., 2017) and their deep counterparts (Blomqvist et al., 2018) are compared.

Ensuing work by Salimbeni and Deisenroth (2017) on the topic of scaling DGPs introduced an alternative approximation based on the variational inducing points framework. This work matches the scalability of our random-features approach, while also leveraging the alternative properties associated with approximations based on inducing points. In a landmark experiment for DGPs, a distributed variation of their model constructed similarly to the set-up described in Section 5.5.4 was evaluated on a dataset having over a billion data-points. There has also been renewed interest in investigating the connection between neural network architectures and (deep) GPs. Prior to our contribution, in an appendix to their work on interpreting dropout as a Bayesian approximation, Gal and Ghahramani (2016) illustrated how MC dropout results in the construction of a Bayesian neural network that is equivalent to a DGP having an arbitrary kernel constructed via network weights. Other recent works expanding on this connection include Louizos and Welling (2016), Matthews et al. (2018), and Lee et al. (2018).

The DGP approximation based on random feature expansions presented in this chapter has itself also served as either the basis or a major point of reference for several extensions and applications of scalable DGPs. These include adaptation to variational auto-encoders for novelty detection (Domingues et al., 2018), recurrent modelling (Föll et al., 2017), problems with constrained function dynamics (Lorenzi and Filippone, 2018), and application to neuroimaging (Nader et al., 2018).

5.7 Conclusion

In this chapter, we expanded our view of GPs to frame them within the wider landscape of deep leaning methods. This had implications on both the degree of scalability expected of the GP approximations investigated in this setting, as well as their overall modelling capacity and flexibility. By way of our presentation of the AutoGP framework, we have jointly investigated three complementary areas of research within the GP community, and demonstrated how unifying these directions of improvement enables GPs to perform com-

petitively with more widely-used neural network architectures. The results obtained with this approach also emphasise how the application of GPs to truly large datasets is no longer an insuperable task, but should motivate further work on improving the quality of models targeting this goal. More significantly, in this chapter we proposed a novel formulation of DGPs that is not only faster, but also frequently outperforms related state-of-the-art methods. The evaluation and results obtained for both the Airline dataset and the MNIST-8M digit recognition problem are particularly impressive since such large datasets had previously been considered to be beyond the computational scope of DGPs. We perceive this to be a considerable step forward in the direction of igniting more interest and encouraging the widespread use of DGPs by both practitioners and the broader machine learning community.

Chapter 6

Conclusion

The models and techniques presented in this thesis are unified by the overarching goal of adapting Gaussian processes and their deep counterparts to the requirements and constraints posed by large datasets and complex problems in the big data regime. We conclude this thesis by summarising the principal themes and contributions presented in the preceding chapters, with particular emphasis on their significance in the context of complementary work in this direction of research. This is followed by a brief outlook on possible avenues for future work where we indicate how one might go about achieving these objectives. Finally, we close with an introspective commentary on the unprecedented rate at which machine learning is developing, highlighting the potential pitfalls that research carried out at such an accelerated pace is susceptible to.

6.1 Themes and Contributions

In this thesis, we primarily investigated the following themes in relation to Gaussian processes:

- **Exact inference on a computational budget [Chapter 3]**

  Developing scalable approximations for Gaussian process models has been a long-standing research topic for which several techniques and methodologies have been developed. In spite of their success in handling the intractability associated with GPs, such approximations often imply making changes to the original model, and trade off superior computational speed-up with poorer predictive performance. In

Chapter 3, we presented an interpretation of GP regression and classification that exploits accelerated solutions to linear systems obtained by way of preconditioned conjugate gradient in order carry out exact GP inference on a computational budget. Contrary to traditional sparse GP approximations, this approach achieves tractability by developing unbiased estimates of the computationally expensive algebraic computations that pose a bottleneck to scalability; in the limit of unbounded computational resources, the results obtained using such a formulation will be exact. Our experimental evaluation indicates that GP inference enhanced with preconditioning improves upon the predictive performance of commonly-used approximations, while also circumventing the computational and storage intractabilities associated with carrying out Cholesky decompositions of large matrices. Only very recently, Wang et al. (2019) have also shown how preconditioning can be exploited in a scheme for training exact GPs on datasets having over a million data-points. This is a breakthrough result for GPs which gives further credence to the methodology we proposed and advocated for in this thesis.

- **Numerical uncertainty in algebraic approximation [Chapter 4]**

  Although approximations in the vein of the aforementioned preconditioning scheme are particularly appealing under the assumption of unrestricted computational budgets, such methodologies must always be considered in light of the potential loss of precision arising from estimating algebraic operations. In the spirit of probabilistic numerics, in Chapter 4 we explored the possibility of quantifying the computational uncertainty associated with such approximations when featured in GP inference and training. We contributed towards this goal by developing a Bayesian approach for approximating the log determinant of large matrices; this contribution extends beyond GPs and also has scope in other domains where estimating the log determinant of large matrices is required. Nonetheless, in spite of the strong motivation for incorporating probabilistic numerics in the deployment of GPs, the results presented in this thesis indicate that the state-of-the-art in this field may not yet be sufficiently robust for attaining such a complex goal.

- **Adaptability to the big data regime [Chapter 5]**

  Whereas most GP approximations are primarily intended for medium-sized datasets, the majority of deep learning techniques are directly applicable to datasets having millions or even billions of data-points. The AutoGP model affronts this goal by jointly exploring three complementary features of GPs that improve their competi-

tiveness with widely-used deep learning models. In particular, AutoGP combines a superior variational approximation of the posterior, flexible kernel design, and optimisation with respect to a leave-one-out objective, in order to obtain state-of-the-art results for GPs on benchmark datasets such as MNIST. The performance of the model was also evaluated on a multi-class classification dataset having over 8 million observations, which is far beyond the scale for which GP inference had previously been considered feasible.

- **Probabilistic deep learning [Chapter 5]**

  The final contribution presented in this thesis was a novel DGP approximation that overcomes the issues and limitations that had previously hindered the application of DGPs to larger datasets. Our proposition employs random feature expansions at each layer of the DGP in order to obtain a model architecture that closely resembles a deep neural network. Aside from being faster than other DGP approximations, our proposal consistently performs better than competing techniques, and improves upon a shallow GP baseline. By demonstrating that DGPs can be applied to very large datasets, and showing how the architecture of such models can extend beyond just a few layers, we have in turn widened the scope and variety of problems to which such models can be applied.

6.2 Future Work

Beyond the extensive discussion featured in this thesis, the themes explored in this body of work not only motivate immediate extensions for improvement, but also set the foundations for broader long-term objectives. In this section, we expand upon the directions for future work which we believe to be particularly pertinent to ongoing developments in both the theoretical and practical aspects of machine learning using GPs. We partition this discussion into the overarching themes of (i) carrying out GP inference and training on a computational budget; and (ii) reinforcing the position of DGPs in relation to other deep learning techniques.

6.2.1 GP Inference on a Budget

The use of preconditioning for accelerating GP inference has recently been used in GPyTorch (Gardner et al., 2018), an open-source library for constructing GPs developed in

PyTorch (Paszke et al., 2017). This is combined with a black-box matrix-matrix multiplication algorithm which lowers the asymptotic complexity of the associated iterative solvers from $\mathcal{O}\left(N^3\right)$ to $\mathcal{O}\left(N^2\right)$. However, the authors limit their discussion to a generic preconditioner that does not necessarily capture the inherent structure expected of kernel matrices; this encourages a more in-depth investigation into whether the suite of preconditioners developed in Chapter 3 could be suitable for use within this framework. The theory developed in Geoga et al. (2018) for using hierarchical matrices to speed up computation of both the GP marginal likelihood and its derivatives should also be considered for not only developing new preconditioners, but also as an alternative GP formulation.

Quantifying the numerical uncertainty associated with approximating linear algebra also remains an important topic for future work to develop. As illustrated by the results presented in Chapter 4, there is certainly more work to be done for improving both the approximation quality and uncertainty calibration of probabilistic numerical linear solvers and the Bayesian log determinant estimation presented in this thesis. In a departure from the Bayesian perspective we have advocated for in this thesis, a sensible progression in this regard would be to additionally consider alternative probabilistic, but not necessarily Bayesian, interpretations of linear algebra, as found in Boutsidis et al. (2017) and Ubaru et al. (2017), for which a measure of confidence accompanying the approximation can also be derived. An insightful analysis contemplating the suitability of Bayesian inference for carrying out linear algebra was very recently presented in Briol et al. (2018).

6.2.2 GPs as an alternative to Neural Networks

There are several immediate extensions that can be considered for the DGP approximation presented in Chapter 5. The first involves introducing the local reparameterisation trick developed in Kingma et al. (2015), which is intended to further reduce uncertainty in the estimation of gradients as well as accelerate overall computation. Superior initialisation strategies should also be considered in order to improve the stability of optimising the model. The results obtained on higher-dimensional datasets also strongly suggest that approximations such as Fastfood (Le et al., 2013) and orthogonal random features (Yu et al., 2016) could be instrumental in the interest of using more random features in their construction. The notion of using low-precision representations of random features, as featured in Zhang et al. (2018), could also improve speed while simultaneously permitting the use of more random features. On another note, incorporating convolutional structure to emulate the architectures explored in van der Wilk et al. (2017), Kumar et al. (2018), and Blomqvist et al. (2018) is essential for obtaining superior performance on complex image classification

tasks such as CIFAR and ImageNet, where the proposed model currently falters. Finally, the continued resilience of Monte Carlo dropout (Gal and Ghahramani, 2016) in the advent of alternative formulations of Bayesian neural networks also encourages their connection to GPs, and especially DGPs, to be explored further.

While the concepts introduced in this thesis are agnostic to specific datasets or applications, the experimental evaluation for both AutoGP and the proposed DGP approximation are heavily influenced by the classification benchmarks prevalent in the deep learning community. Exploring alternative problems would not only inspire and reveal different modelling requirements than those we are accustomed to, but also further broaden the scope and appeal of DGPs to researchers and practitioners alike. As a motivating example, in Cutajar et al. (2018) we demonstrate how DGPs can be interpreted for the purpose of multi-fidelity modelling, whereby DGP layers are treated as fidelity levels, and a variational inference scheme is used to propagate uncertainty across them. The utility of DGPs in view of their theoretical properties was recently rigorously studied in Dunlop et al. (2018), which should serve as a guide for future work in this domain.

6.3 This ain't Science, it's an Arms Race[1]

Twenty years ago, the late Prof. Sir David MacKay concluded a general introduction to GPs with a discussion on how such methods compare to neural networks, posing the memorable questions *"Have we thrown the baby out with the baby water? How can Gaussian processes possibly replace neural networks? What is going on?"* (MacKay, 1998). This commentary was ahead of its time in the way it predicted how the rigid assumptions characterising GPs could prevent them from properly matching the performance of more flexible deep learning techniques. In essence, much of recent work published in relation to GPs, including the models introduced in the latter parts of this thesis, has predominantly involved playing 'catch-up' with more advanced deep learning techniques by reinterpreting those models with a Bayesian perspective.

In many ways, this urgency to compete with deep learning techniques is also reflective of the overall state of machine learning research. At the time of its publication, the work presented in Chapter 3 was prized for applying GPs to datasets with tens of thousands of observations without relying on traditional approximations. This was in line with the long-standing belief that GPs are primarily data-efficient models intended for tasks requiring high precision or having considerable risk. Conversely, the models presented in

[1]The title of this section is a play on the hit song *"This Ain't a Scene, it's an Arms Race"*.

Chapter 5 were developed in response to concerns that GPs were no longer appealing in the advent of more competitive neural network architectures. Indeed, our work convincingly demonstrates that the connections between GPs and neural networks are key to modernising GPs; in a first for GPs and especially DGPs, we showed how the proposed model architectures can be applied to complex multi-class classification problems having over 8 million data-points. Notwithstanding, in less than a year after both works were published, the application of GPs to massive datasets has already become a de facto accomplishment, and attention has instead been diverted towards directly competing with more advanced deep learning methods through incorporating convolutions, recurrent structure, and the list goes on.

In a twist on the quoted remarks setting the tone for this conclusion, we opine that beyond fuelling the ongoing 'arms race' to outperform existing deep learning techniques on problems where such methods already work well, ongoing and future work on GPs should be steered more towards problems where deep learning models currently face a greater hurdle. Alongside probabilistic numerics, which was directly featured in this thesis, we believe that GPs will also play an important role in emerging fields such as interpretable machine learning (Lipton, 2018) and data-efficient reinforcement learning (Kamthe and Deisenroth, 2018), among others.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. (2017). Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 18:82:1–82:37.

Anitescu, M., Chen, J., and Wang, L. (2012). A matrix-free approach for solving the parametric Gaussian process maximum likelihood problem. *SIAM Journal on Scientific Computing*, 34(1):A240–A262.

Asuncion, A. and Newman, D. J. (2007). UCI machine learning repository.

Avron, H., Clarkson, K. L., and Woodruff, D. P. (2017a). Faster kernel ridge regression using sketching and preconditioning. *SIAM Journal of Matrix Analysis Applications*, 38(4):1116–1138.

Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., and Zandieh, A. (2017b). Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *34th International Conference on Machine Learning (ICML)*, pages 253–262.

Avron, H. and Toledo, S. (2011). Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):8:1–8:34.

Awad, E., Levine, S., Kleiman-Weiner, M., Dsouza, S., Tenenbaum, J. B., Shariff, A., Bonnefon, J.-F., and Rahwan, I. (2018). Blaming humans in autonomous vehicle accidents: Shared responsibility across levels of automation. *arXiv preprint arXiv:1803.07170*.

Axelsson, O. (1994). *Iterative Solution Methods*. Cambridge University Press, New York, NY, USA.

Bartels, S., Cockayne, J., Ipsen, I. C. F., and Hennig, P. (2018). Probabilistic linear solvers: A unifying view. *arXiv preprint arXiv:1810.03398*.

Bartels, S. and Hennig, P. (2016). Probabilistic approximate least-squares. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 676–684.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1533–1541.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2017). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18:153:1–153:43.

Benoît, C. (1924). Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieur a celui des inconnues. — application de la méthode a la résolution d'un système defini d'équations linéaires. *Bulletin géodésique*, 2(1):67–77.

Bimbraw, K. (2015). Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 191–198.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Blomqvist, K., Kaski, S., and Heinonen, M. (2018). Deep convolutional Gaussian processes. *arXiv preprint arXiv:1810.03052*.

Bonilla, E. V., Krauth, K., and Dezfouli, A. (2016). Generic inference in latent Gaussian process models. *arXiv preprint arXiv:1609.00577*.

Bonnefon, J.-F., Shariff, A., and Rahwan, I. (2016). The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576.

Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, Inc., New York, NY, USA.

Boutsidis, C., Drineas, P., Kambadur, P., Kontopoulou, E.-M., and Zouzias, A. (2017). A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533:95–117.

Briol, F.-X., Oates, C. J., Girolami, M., Osborne, M. A., and Sejdinovic, D. (2018). Rejoinder for "Probabilistic integration: A role in statistical computation?". *arXiv preprint arXiv:1811.10275*.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886.

Bui, T. D., Hernández-Lobato, D., Hernández-Lobato, J. M., Li, Y., and Turner, R. E. (2016). Deep Gaussian processes for regression using approximate expectation propagation. In *33rd International Conference on Machine Learning (ICML)*, pages 1472–1481.

Bui, T. D., Yan, J., and Turner, R. E. (2017). A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 18:104:1–104:72.

Chalupka, K., Williams, C. K. I., and Murray, I. (2013). A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14(1):333–350.

Chen, J., Monga, R., Bengio, S., and Jozefowicz, R. (2016). Revisiting distributed synchronous SGD. In *Workshop Track, 4th International Conference on Learning Representations (ICLR)*.

Chilimbi, T. M., Suzue, Y., Apacible, J., and Kalyanaraman, K. (2014). Project Adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation OSDI*, pages 571–582.

Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22 (NeurIPS)*, pages 342–350.

Cockayne, J., Oates, C. J., and Girolami, M. A. (2018). A Bayesian conjugate gradient method. *arXiv preprint arXiv:1801.05242*.

Csató, L. and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668.

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. (2017). Random feature expansions for deep Gaussian processes. In *34th International Conference on Machine Learning (ICML)*, pages 884–893.

Cutajar, K., Osborne, M. A., Cunningham, J. P., and Filippone, M. (2016). Preconditioning kernel matrices. In *33rd International Conference on Machine Learning (ICML)*, pages 2529–2538.

Cutajar, K., Pullin, M., Damianou, A., Lawrence, N. D., and Gonzalez, J. (2018). Deep Gaussian processes for multi-fidelity modeling. In *Bayesian Deep Learning Workshop, Advances in Neural Information Processing Systems 31 (NeurIPS)*.

Dai, Z., Damianou, A., González, J., and Lawrence, N. D. (2016). Variational auto-encoded deep Gaussian processes. In *4th International Conference on Learning Representations (ICLR)*.

Damianou, A. (2015). *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, University of Sheffield.

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In *16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 207–215.

Davies, A. (2014). *Effective Implementation of Gaussian Process Regression for Machine Learning*. PhD thesis, University of Cambridge.

Davis, T. A. and Hu, Y. (2011). The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1.

de Finetti, B. (1974). *Theory of Probability*. Wiley, New York. First of two volumes translated from *Teoria Delle probabilità*, published 1970. The second volume appeared under the same title in 1975.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems 26 (NeurIPS)*, pages 2148–2156.

Dezfouli, A. and Bonilla, E. V. (2015). Scalable inference for Gaussian process models with black-box likelihoods. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 1414–1422.

Domingues, R., Michiardi, P., Zouaoui, J., and Filippone, M. (2018). Deep Gaussian process autoencoders for novelty detection. *Machine Learning, Special Issue of the ECML PKDD 2018 Journal Track, ISSN: 0885-6125*.

Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. (2017). Scalable log determinants for Gaussian process kernel learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6330–6340.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dunlop, M. M., Girolami, M. A., Stuart, A. M., and Teckentrup, A. L. (2018). How deep are deep Gaussian processes? *Journal of Machine Learning Research*, 19(54):1–46.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

Duvenaud, D., Rippel, O., Adams, R. P., and Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 202–210.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Ekenberg, L. and Thorbiörnson, J. (2001). Second-order decision analysis. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(01):13–37.

Evans, T. W. and Nair, P. B. (2018). Exploiting structure for fast kernel learning. In *SIAM International Conference on Data Mining, (SDM)*, pages 414–422.

Filippone, M. and Engler, R. (2015). Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the unbiased linear system solver (ULISSE). In *32nd International Conference on Machine Learning (ICML)*, pages 1015–1024.

Filippone, M., Zhong, M., and Girolami, M. (2013). A comparative evaluation of stochastic-based inference methods for Gaussian process models. *Machine Learning*, 93(1):93–114.

Fitzsimons, J. K., Cutajar, K., Filippone, M., Osborne, M. A., and Roberts, S. J. (2017a). Bayesian inference of log determinants. In *33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Fitzsimons, J. K., Granziol, D., Cutajar, K., Osborne, M. A., Filippone, M., and Roberts, S. J. (2017b). Entropic trace estimates for log determinants. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML/PKDD*, pages 323–338.

Fitzsimons, J. K., Osborne, M. A., Roberts, S. J., and Fitzsimons, J. F. (2018). Improved stochastic trace estimation using mutually unbiased bases. In *34th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Flaxman, S., Wilson, A., Neill, D., Nickisch, H., and Smola, A. (2015). Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods. In *32nd International Conference on Machine Learning (ICML)*, pages 607–616.

Föll, R., Haasdonk, B., Hanselmann, M., and Ulmer, H. (2017). Deep recurrent Gaussian process with variational sparse spectrum approximation. *arXiv preprint arXiv:1711.00799*.

Frey, C. B. and Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 114:254 – 280.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning (ICML)*, pages 1050–1059.

Gal, Y. and Turner, R. (2015). Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *32nd International Conference on Machine Learning (ICML)*, pages 655–664.

Gal, Y., van der Wilk, M., and Rasmussen, C. E. (2014). Distributed variational inference in sparse Gaussian process regression and latent variable models. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 3257–3265.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 7587–7597.

Geoga, C. J., Anitescu, M., and Stein, M. L. (2018). Scalable Gaussian process computations using hierarchical matrices. *arXiv preprint arXiv:1808.03215*.

Gershgorin, S. (1931). Uber die Abgrenzung der Eigenwerte einer Matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(3):749–754.

Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society*, 371(1984):20110553.

Gilboa, E., Saatçi, Y., and Cunningham, J. P. (2013). Scaling multidimensional Gaussian processes using projected additive approximations. In *30th International Conference on Machine Learning (ICML)*, pages 454–461.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix computations*. The Johns Hopkins University Press, 3rd edition.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*, pages 2348–2356.

Gray, R. M. (1990). *Entropy and Information Theory*. Springer-Verlag, Berlin, Heidelberg.

Halko, N., Martinsson, P., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.

Han, I., Malioutov, D., and Shin, J. (2015). Large-scale log-determinant computation through stochastic Chebyshev expansions. In *32nd International Conference on Machine Learning (ICML)*, pages 908–917.

Henao, R. and Winther, O. (2012). Predictive active set selection methods for Gaussian processes. *Neurocomputing*, 80:10–18.

Hennig, P. (2015). Probabilistic interpretation of linear solvers. *SIAM Journal on Optimization*, 25(1):234–260.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179).

Hensman, J., Durrande, N., and Solin, A. (2017). Variational fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18:151:1–151:52.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *29th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Hensman, J. and Lawrence, N. D. (2014). Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*.

Hensman, J., Matthews, A. G. d. G., Filippone, M., and Ghahramani, Z. (2015a). MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 1648–1656.

Hensman, J., Matthews, A. G. d. G., and Ghahramani, Z. (2015b). Scalable variational Gaussian process classification. In *18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 351–360.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *32nd International Conference on Machine Learning (ICML)*, pages 1861–1869.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436.

Hoang, Q. M., Hoang, T. N., and Low, K. H. (2017). A generalized stochastic variational Bayesian hyperparameter learning framework for sparse spectrum Gaussian process regression. In *31st Conference on Artificial Intelligence (AAAI)*, pages 2007–2014.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. W. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.

Huszar, F. and Duvenaud, D. (2012). Optimally-weighted herding is Bayesian quadrature. In *28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 377–386.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Kaasschieter, E. F. (1988). Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied mathematics*, 24(1-2):265–275.

Kamthe, S. and Deisenroth, M. P. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. In *21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1701–1710.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5580–5590.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 2575–2583.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*.

Kom Samo, Y.-L. and Roberts, S. (2015). Generalized spectral kernels. *arXiv preprint arXiv:1506.02236*.

Krauth, K., Bonilla, E. V., Cutajar, K., and Filippone, M. (2017). AutoGP: Exploring the capabilities and limitations of Gaussian process models. In *33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Krzywinski, M. and Altman, N. (2013). Importance of being uncertain. *Nature Methods*, 10:809 – 810.

Kumar, V., Singh, V., Srijith, P. K., and Damianou, A. (2018). Deep Gaussian processes with convolutional kernels. *arXiv preprint arXiv:1806.01655*.

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704.

Lawrence, N. D. and Moore, A. J. (2007). Hierarchical Gaussian process latent variable models. In *24th International Conference on Machine Learning (ICML)*, pages 481–488.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881.

Le, Q. V., Sarlós, T., and Smola, A. J. (2013). Fastfood - computing Hilbert space expansions in loglinear time. In *30th International Conference on Machine Learning (ICML)*, pages 244–252.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. (2018). Deep neural networks as Gaussian processes. In *6th International Conference on Learning Representations (ICLR)*.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. (2014). Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 19–27.

Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM*, 61(10):36–43.

Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2018). When Gaussian process meets big data: A review of scalable GPs. *arXiv preprint arXiv:1807.01065*.

Loosli, G., Canu, S., and Bottou, L. (2007). Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA.

Lorenzi, M. and Filippone, M. (2018). Constraining the dynamics of deep probabilistic models. In *35th International Conference on Machine Learning (ICML)*, pages 3233–3242.

Louizos, C. and Welling, M. (2016). Structured and efficient variational deep learning with matrix Gaussian posteriors. In *33rd International Conference on Machine Learning (ICML)*, pages 1708–1716.

MacKay, D. J. C. (1991). Bayesian interpolation. *Neural Computation*, 4:415–447.

MacKay, D. J. C. (1996). Hyperparameters: optimize, or integrate out? In *Maximum Entropy and Bayesian Methods*, pages 43–60. Kluwer.

MacKay, D. J. C. (1998). Introduction to Gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*, chapter 28, pages 343–355. Cambridge university press.

Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. (2014). Convolutional kernel networks. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 2627–2635.

Matthews, A. G. d. G., Hensman, J., Turner, R. E., and Ghahramani, Z. (2016). On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 231–239.

Matthews, A. G. d. 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)*.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18:40:1–40:6.

Minka, T. P. (2001). *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology.

Morris, M. D. (2004). The design and analysis of computer experiments. *Journal of the American Statistical Association*, 99(468):1203–1204.

Murray, I., Adams, R. P., and MacKay, D. J. C. (2010). Elliptical slice sampling. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 541–548.

Nader, C. A., Ayache, N., Robert, P., and Lorenzi, M. (2018). Alzheimer's disease modelling and staging through independent Gaussian process analysis of spatio-temporal brain changes. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications - First International Workshops MLCN, DLF, and iMIMIC, Held in Conjunction with MICCAI*, pages 3–14.

Neal, R. M. (1992). Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems 5 (NeurIPS)*, pages 475–482.

Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078.

Notay, Y. (2000). Flexible conjugate gradients. *SIAM Journal on Scientific Computing*, 22(4):1444–1460.

O'Hagan, A. (1991). Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, 29:245–260.

Opper, M. and Archambeau, C. (2009). The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *Autodiff Workshop, Advances in Neural Information Processing Systems 31 (NeurIPS)*.

Petrone, G. (2011). *Optimization under Uncertainty: theory, algorithms and industrial applications*. PhD thesis.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. (2018). Constant-time predictive distributions for Gaussian processes. In *35th International Conference on Machine Learning (ICML)*, pages 4111–4120.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20 (NeurIPS)*, pages 1177–1184.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 814–822.

Rasmussen, C. E. and Ghahramani, Z. (2000). Occam's razor. In *Advances in Neural Information Processing Systems 13 (NeurIPS)*, pages 294–300.

Rasmussen, C. E. and Ghahramani, Z. (2002). Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15 (NeurIPS)*, pages 489–496.

Rasmussen, C. E. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Remes, S., Heinonen, M., and Kaski, S. (2017). Non-stationary spectral kernels. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 4642–4651.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *31st International Conference on Machine Learning (ICML)*, pages 1278–1286.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Roberts, S., Osborne, M. A., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 371:20110550.

Rudi, A. and Rosasco, L. (2017). Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 3218–3228.

Rudin, W. (1990). *Fourier Analysis on Groups*. A Wiley-interscience publication. Wiley.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition.

Saatçi, Y. (2012). *Scalable inference for structured Gaussian process models*. PhD thesis, University of Cambridge.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6655–6659.

Salimbeni, H. and Deisenroth, M. P. (2017). Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 4591–4602.

Salimbeni, H., Eleftheriadis, S., and Hensman, J. (2018). Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 689–697.

Schölkopf, B. (1997). *Support vector learning*. PhD thesis, Berlin Institute of Technology.

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *9th International Workshop on Artificial Intelligence and Statistics*.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Shen, Z., Heinonen, M., and Kaski, S. (2018). Harmonizable mixture kernels with variational Fourier features. *arXiv preprint arXiv:1810.04416*.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR) - Volume 2*, pages 958–.

Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13 (NeurIPS)*, pages 619–625.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18 (NeurIPS)*, pages 1257–1264.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. In *11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 524–531.

Srinivasan, B. V., Hu, Q., Gumerov, N. A., Murtugudde, R., and Duraiswami, R. (2014). Preconditioned Krylov solvers for kernel regression. *arXiv preprint arXiv:1408.1237*.

Sundararajan, S. and Keerthi, S. S. (2001). Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural Computation*, 13(5):1103–1118.

Sundararajan, S. and Keerthi, S. S. (2012). Predictive approaches for Gaussian process classifier model selection. *arXiv preprint arXiv:1206.6038*.

Tan, L. S., Ong, V. M., Nott, D. J., and Jasra, A. (2016). Variational inference for sparse spectrum Gaussian process regression. *Statistics and Computing*, 26(6):1243–1261.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 567–574.

Ton, J.-F., Flaxman, S., Sejdinovic, D., and Bhatt, S. (2018). Spatial mapping with Gaussian processes and nonstationary Fourier features. *Spatial Statistics*, 28:59 – 78.

Ubaru, S., Chen, J., and Saad, Y. (2017). Fast estimation of tr(f(a)) via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 2845–2854.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., and Vehtari, A. (2013). GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14(1):1175–1179.

Vehtari, A., Mononen, T., Tolvanen, V., Sivula, T., and Winther, O. (2016). Bayesian leave-one-out cross-validation approximations for Gaussian latent variable models. *Journal of Machine Learning Research*, 17:103:1–103:38.

Wahba, G. (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia.

Wang, K. A., Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. (2019). Exact Gaussian processes on a million data points. *arXiv pre-print arXiv:1903.08114*.

Wang, Y., Brubaker, M., Chaib-Draa, B., and Urtasun, R. (2016). Sequential inference for deep Gaussian process. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 694–703.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *28th International Conference on Machine Learning (ICML)*, pages 681–688.

Williams, C., Rasmussen, C., Schwaighofer, A., and Tresp, V. (2002). Observations on the Nyström method for Gaussian process prediction. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13 (NeurIPS)*, pages 682–688.

Wilson, A. G. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *30th International Conference on Machine Learning (ICML)*, pages 1067–1075.

Wilson, A. G., Gilboa, E., Nehorai, A., and Cunningham, J. P. (2014). Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 3626–3634.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016a). Deep kernel learning. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 370–378.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 2594–2602.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *29th International Conference on Machine Learning (ICML)*, pages 1139–1146.

Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *32nd International Conference on Machine Learning (ICML)*, pages 1775–1784.

Wu, J. and Frazier, P. I. (2017). Continuous-fidelity Bayesian optimization with knowledge gradient. In *BayesOpt Workshop, Advances in Neural Information Processing Systems 30 (NeurIPS)*.

Wu, J., Poloczek, M., Wilson, A. G., and Frazier, P. (2017). Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5267–5278.

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)*, pages 1975–1983. Curran Associates, Inc.

Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. (2017). Advances in variational inference. *arXiv preprint arXiv:1711.05597*.

Zhang, J., May, A., Dao, T., and Ré, C. (2018). Low-precision random Fourier features for memory-constrained kernel approximation. *arXiv preprint arXiv:1811.00155*.

Zhu, W., Miao, J., Qing, L., and Chen, X. (2015). Deep trans-layer unsupervised networks for representation learning. *arXiv preprint arXiv:1509.08038*.

Appendix A

Conjugate Gradient Algorithm

One of the primary contributions of this thesis involves the design and selection of preconditioners which are well suited to kernel matrices in order to accelerate GP training and inference by way of preconditioned conjugate gradient. In our evaluation, we consistently assessed the effectiveness of the proposed methodology using plain conjugate gradient as a baseline over which performance improvements should be measured. A description of the motivation and purpose of conjugate gradient is already given in Chapter 3, and here we supplement that commentary by providing the full pseudocode for this algorithm.

Algorithm 3 Conjugate Gradient Algorithm, adapted from Golub and Van Loan (1996)

Input: Matrix A, vector $\mathbf{b}$, convergence threshold ε, initial estimate $\mathbf{z}_0$, maximum number of iterations T

$\mathbf{r}_0 = \mathbf{b} - A\mathbf{z}_0$; $\mathbf{s}_0 = \mathbf{r}_0$

for $t = 0 : T$ do

$\alpha_t = \frac{\mathbf{r}_t^\top \mathbf{r}_t}{\mathbf{s}_t^\top A \mathbf{s}_t}$

$\mathbf{z}_{t+1} = \mathbf{z}_t + \alpha_t \mathbf{s}_t$

$\mathbf{r}_{t+1} = \mathbf{r}_t - \alpha_t A \mathbf{s}_t$

if $\|\mathbf{r}_{t+1}\| < \varepsilon$ then

return $\mathbf{z}_{t+1}$

end if

$\beta_t = \frac{\mathbf{r}_{t+1}^\top \mathbf{r}_{t+1}}{\mathbf{r}_t^\top \mathbf{r}_t}$

$\mathbf{s}_{t+1} = \mathbf{r}_{t+1} + \beta_t \mathbf{s}_t$

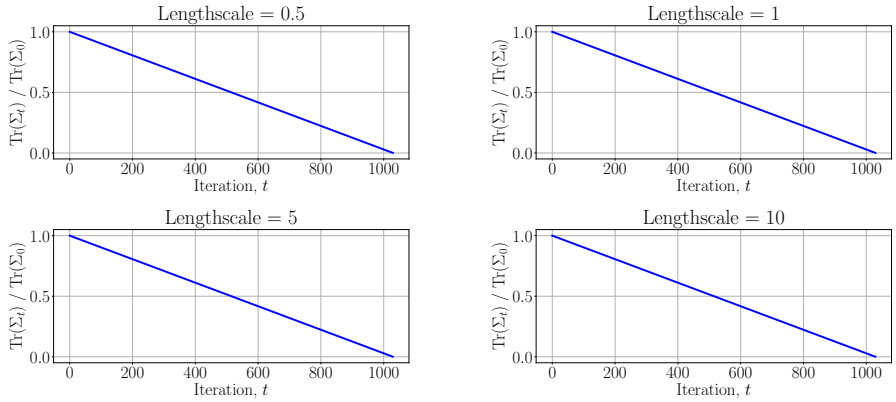
end for

return $\mathbf{z}_{t+1}$

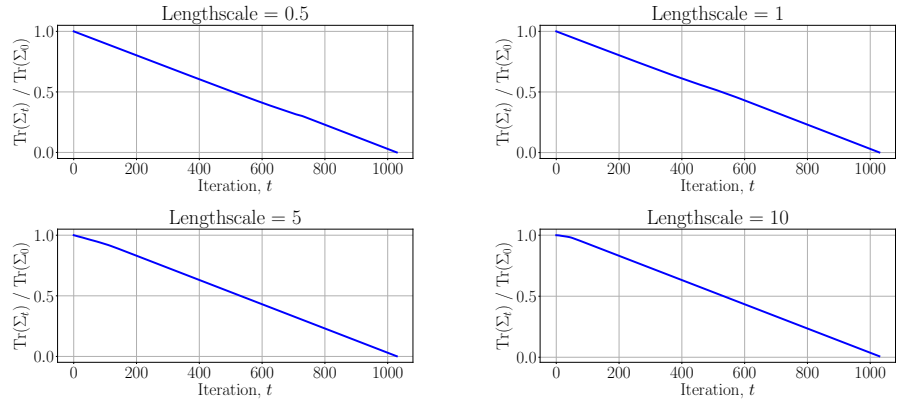
Appendix B

Continuation of Probabilistic Numerics Evaluation

In Section 4.4.3 of Chapter 4, we investigated the suitability of probabilistic linear solvers for estimating the solution of linear systems involving kernel matrices constructed using a variety of parameter settings. Following up on the assessment of the uncertainty of BCG for the White Wine dataset, in Figure B.1 we present the results for the same experiment carried out using the Concrete dataset instead. The results mirror those obtained for the other dataset, whereby the decrease in uncertainty relative to the prior is linear in the number of iterations; as before, these results would be difficult to incorporate effectively within a pipelined probabilistic numerics scheme.



(a) Uncertainty quality of solution-based inference with prior $\Sigma_0 = I_N$.



(b) Uncertainty quality of solution-based inference with prior $\Sigma_0 = P^{-1}$.

Fig. B.1 Evaluation of uncertainty obtained from probabilistic linear solvers for kernel matrices evaluated over the **Concrete** dataset. We plot the summation of variances across the diagonal of the posterior covariance, $\mathrm{Tr}\left(\Sigma_t\right)$, normalised by the trace of the prior covariance I_N or P^{-1}.

Appendix C

Further DGP Experiments

In Section 5.4.5 of Chapter 5, we outlined the different strategies for treating the random mappings $\boldsymbol{\Omega}$ in our DGP approximation. In particular, we distinguished between fixing them or treating them variationally, where we observed that the constructed DGP model appears to work best when these are treated variationally while fixing the randomness in their computation throughout the learning process (VAR-FIXED). In Figures C.1 and C.2, we compare the three proposed approaches on the complete set of datasets reported in the main document for one and two hidden layers, respectively. Once again, we confirm that the performance obtained using the VAR-FIXED strategy yields consistently better results than the alternatives. This is especially pertinent to the classification datasets, where the obtained error rate is markedly superior. However, the variation of the model constructed with the arc-cosine kernel and optimised using VAR-FIXED appears to be susceptible to some overfitting for higher dimensional datasets (Spam and MNIST), which is expected given that we are optimising several covariance parameters (ARD). This motivates the inclusion of a variational interpretation of the hyperparameters $\boldsymbol{\Theta}$ as future work.

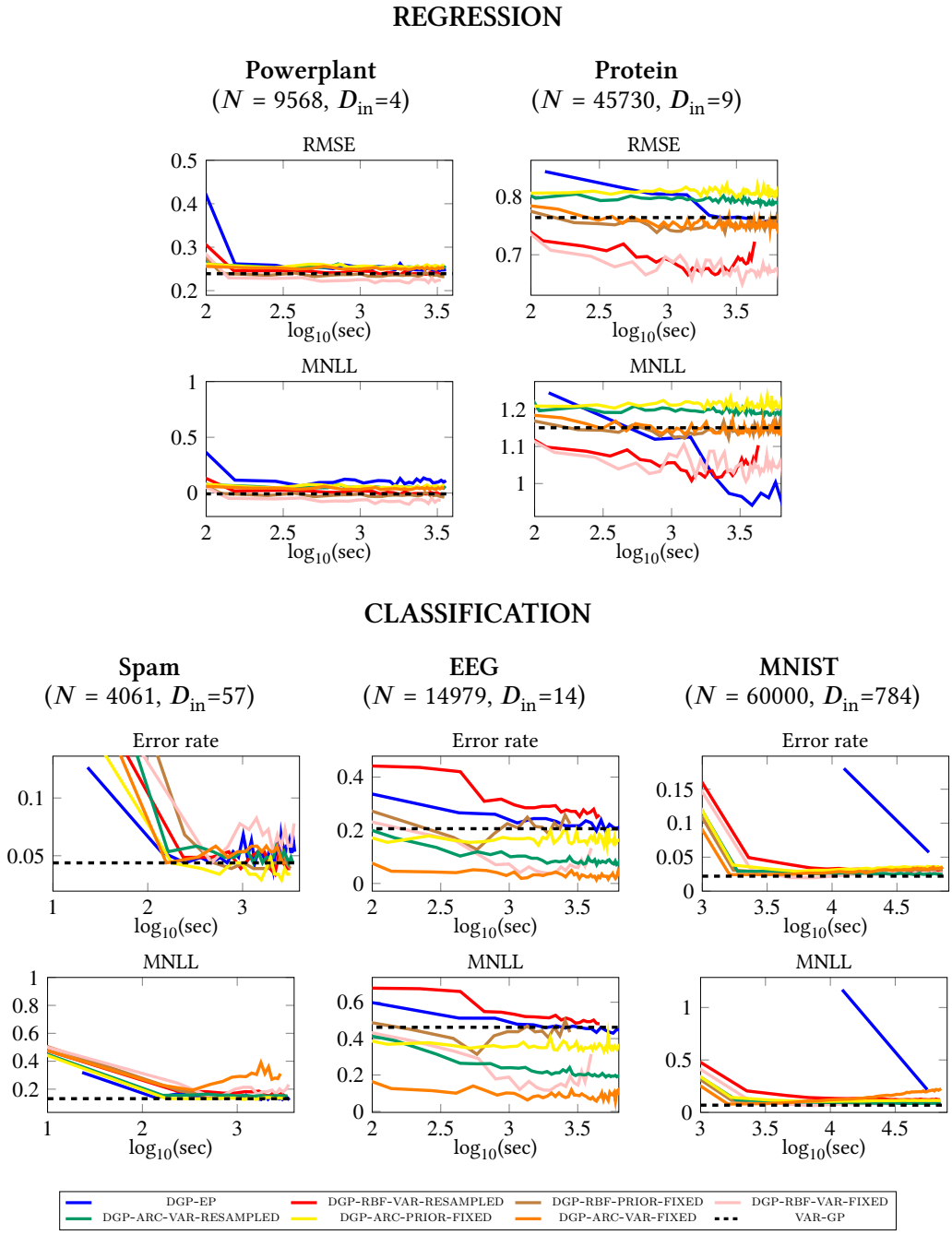


Fig. C.1 Progression of error rate (RMSE in the regression case) and MNLL over time for different optimisation strategies of the DGP-ARC and DGP-RBF models. Results are shown for configurations having 1 hidden layer.

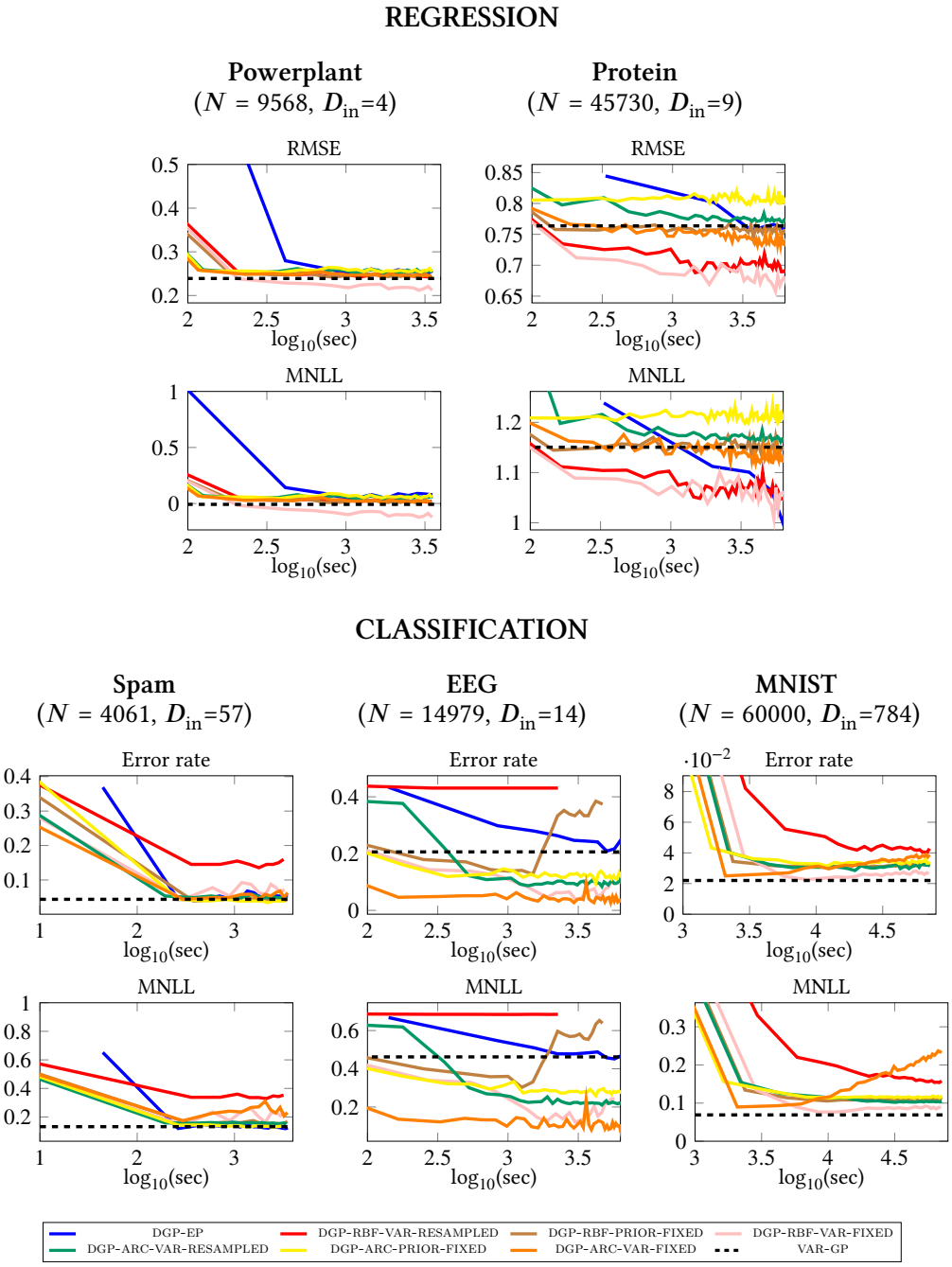


Fig. C.2 Progression of error rate (RMSE in the regression case) and MNLL over time for different optimisation strategies of the DGP-ARC and DGP-RBF models. Results are shown for configurations with 2 hidden layers.