Imperial College London

Imperial College London
Department of Computing

Deep Gaussian Processes: advances in Models and Inference

Hugh Salimbeni

This dissertation is submitted for the degree of
Doctor of Philosophy

June 7, 2020

Declaration

I declare that this thesis is my own work, except where indicated.

Copyright notice

Acknowledgements

FIRSTLY I thank my supervisor Marc Deisenroth for his unwavering support and guidance through the last four years. He has been generous with his time and always on my side. For this I am truly grateful.

At Imperial I have benefited from the support of many other students in the Statistical Machine Learning Group. In particular, conversations over the years with Sanket Kamthe and Steindór Sæundsson have been an important part of my development. In the later part of my PhD I have also benefited from discussions with James Wilson and Alex Terenin. Outside the group but no less important I'd like to thank Pedro Mediano, Matt Lee, Kai Arulkumaran and Kyriacos Nikiforou, Nat Dilokthanakul, Matt Crosby and Marta Garnello for providing such stimulating lunchtime discussion.

The last two years of my PhD have been spent partly at Prowler. I am grateful to Vishal Chatrath for allowing this to happen and for creating such a open and productive research environment. At Prowler I have benefited enormously from the guidance of James Hensman. Many of the ideas in the latter three chapters of this thesis stemmed from discussions with James. Also at Prowler I am fortunate to have found such excellent collaborators in Stefanos Eleftheriadis and Vincent Dutordoir. I'd like also to thank particularly Mark van der Wilk: despite never being a formal collaborator, Mark has had a huge impact on my understanding of many of the ideas in this thesis.

I thank Javier Carnerero Cano for sharing his LaTeX template.

Abstract

HIERARCHICAL models are certainly in fashion these days. It seems difficult to navigate the field of machine learning without encountering 'deep' models of one sort or another. The popularity of the deep learning revolution has been driven by some striking empirical successes, prompting both intense rapture and intense criticism. The criticisms often centre around the lack of model uncertainty, leading to sometimes drastically overconfident predictions. Others point to the lack of a mechanism for incorporating prior knowledge, and the reliance on large datasets. A widely held hope is that a Bayesian approach might overcome these problems.

The deep Gaussian process presents a paradigm for building deep models from a Bayesian perspective. A Gaussian process is a prior for functions. A deep Gaussian process uses several Gaussian process functions and combines them hierarchically through composition (that is, the output of one is the input to the next). The deep Gaussian process promises to capture the compositional nature of deep learning while mitigating some of the disadvantages through a Bayesian approach.

The thesis develops deep Gaussian process modelling in a number of ways. The model is first interpreted differently from previous work, not as a 'hierarchical prior' but as a factorized prior with an hierarchical likelihood. Mean functions are suggested to avoid issues of degeneracy and to aid initialization. The main contribution is a new method of inference that avoids the burden of representing the function values directly through an application of sparse variational inference. This method scales to arbitrarily large data and is shown to work well in practice through experiments.

The use of variational inference recasts (approximate) inference as optimization of Gaussian distributions. This optimization has an exploitable geometry via the natural gradient. The natural gradient is shown to be advantageous for single layer non-conjugate models, and for the (final layer of a) deep Gaussian process model.

Deep Gaussian processes can be a model both for complex associations between variables and complex marginal distributions of single variables. Incorporating noise in the hierarchy leads to complex marginal distribution through the non-linearities of the mappings at each layer. The inference required for noisy variables cannot be handled with sparse methods, as sparse methods rely on correlations between variables, which are absent for noisy variables. Instead, a more direct approach is developed, using an importance weighted variational scheme.

Outline of thesis

- Chapter 1 introduces the Gaussian process model and variational inference, and discusses the limitations of single layer Gaussian processes. The only novel contribution of this chapter is the graphical model and indexing diagram notation for a Gaussian process. This notation is important for later chapters.
- Chapter 2 presents the deep Gaussian process model and proposes a novel method of variational inference. The key idea is to consider inference over the functions themselves, rather than the function values at specific points.
- Chapters 3 extends the natural gradient method to perform fast variational inference in non-conjugate models. This technique is applied to the subsequent chapters.
- Chapter 4 develops a latent variable model, which can be thought of as a two layer Deep GP with a kernel that cannot be represented with the inducing-point approach of Chapter 2. This model therefore requires an alternative approach of inference. Two options are presented and evaluated on a broad range of tasks.
- Chapter 5 extends the latent variable model of Chapter 4 to the multi-layer case, and proposes an novel importance weighted inference scheme for the latent variables. The importance weighted bound is strictly an improvement over the standard variational approach, and achieves better performance in practice.
- Chapter 6 presents some further work from other groups developing and applying ideas presented in this thesis. Chapter 7 discusses limitations and possibilities for further research.

Relationship to published papers

The central content of this thesis appears in four published papers.

- Chapter 2 is based Salimbeni and Deisenroth [2017b] (H. Salimbeni and M. P. Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes, *NeurIPS* 2017). Compared to the published paper this chapter has been largely rewritten to better explain the inference in terms of divergence between processes. The notation has also been completely revised and more emphasis has been placed on the use of a mean function. There are also additional illustrative examples for the 1D case, and figures providing both the graphical model and the evaluation diagram.

- Chapter 3 is based on Salimbeni et al. [2018b] (H. Salimbeni, S. Eleftheriadis, and J. Hensman, Natural Gradients in Practice: Non-conjugate Variational Inference in Gaussian Process Models, *AISTATS* 2018). This chapter closely follows the published paper, but includes new experiments demonstrating the use of natural gradients in the deep model.

- Chapter 4 is based on [Dutordoir et al., 2018] (V. Dutordoir*, H. Salimbeni*, M. P. Deisenroth, and J. Hensman. Gaussian Process Conditional Density Estimation, *NeurIPS* 2018). Some of the experiments were implemented by my collaborator Vincent Dutordior. These experiments are included for completeness and attribution is made clear. The text has been largely re-written for consistency with other chapters. The published paper was written to emphasize the connection to deep generative models, but the version here interprets the model as a deep GP.

- Chapter 5 is based on Salimbeni and Deisenroth [2019] (H. Salimbeni, V. Dutordoir, J. Hensmn and M. P. Deisenroth. Deep Gaussian Processes with Importance Weighted Variational inference. *ICML* 2019). This chapter closely follows the published paper.

The publication Salimbeni et al. [2018a] (H. Salimbeni*, C.-A. Cheng*, B. Boots, and M. Deisenroth. Orthogonally Decoupled Variational Gaussian Processes. *NeurIPS*, 2018) is closely related to Chapter 3, but is not included in this thesis.

Contents

1 Introduction **1**

1.1 Gaussian processes 4

1.2 Variational Inference 7

1.3 Multiple outputs 12

1.4 Learning covariance functions 13

1.5 Limitations of Gaussian processes 17

1.6 Approaches to overcome the limitations of GPs 19

2 Doubly Stochastic Variational Inference for Deep Gaussian Processes **21**

2.1 Related Work 22

2.2 Deep Gaussian Processes 24

2.3 Inference 26

2.4 Results 29

2.5 Conclusion 34

3 Natural Gradients in Practice: Non-Conjugate Variational Inference in Gaussian Process Models **35**

3.1 Introduction 35

3.2 Background 37

3.2.1 Preliminaries 38

3.2.2 Optimization approaches 39

3.2.3 Efficient computation 40

3.2.4 The forward-mode trick 41

3.3 Specific application: sparse Gaussian processes 42

3.4 Natural gradients in practice 43

3.4.1 Deterministic case . . . 44
3.4.2 Stochastic natural gradients . . . 44
3.4.3 Hyperparameters . . . 46
3.4.4 When natural gradients are *essential* . . . 47
3.4.5 Further likelihoods . . . 47
3.4.6 The deep Gaussian process case . . . 48
3.5 Related work . . . 50
3.6 Discussion and conclusion . . . 50

4 Gaussian Process Conditional Density Estimation 52
4.1 Introduction . . . 52
4.2 Inference . . . 54
4.2.1 Gaussian variational distribution for the latent variables . . . 55
4.2.2 Analytically optimal variational distribution for the latent variables . . . 56
4.2.3 Probabilistic linear transformations . . . 57
4.3 Related work . . . 57
4.4 Experiments . . . 58
4.5 Conclusion . . . 64

5 Deep Gaussian Processes with Importance-Weighted Variational Inference 65
5.1 Model . . . 66
5.1.1 Gaussian process (GP) layer . . . 67
5.1.2 Latent-variable (LV) layer . . . 67
5.1.3 Example model: LV-GP-GP . . . 67
5.1.4 Model variants . . . 68
5.1.5 Multiple outputs . . . 68
5.1.6 Mean and covariance functions . . . 69
5.2 Inference in the LV-GP-GP model . . . 69
5.2.1 Variational inference . . . 69
5.2.2 Importance-weighted variational inference . . . 70
5.2.3 Analytic final layer . . . 71
5.2.4 The posterior over the latent variables . . . 73

5.2.5 Further inference details . . . 73
5.3 Results . . . 73
5.3.1 1D example . . . 74
5.3.2 UCI datasets . . . 74
5.4 Related work . . . 77
5.5 Discussion . . . 77
5.5.1 Limitations . . . 78
5.6 Conclusion . . . 78

6 Impact **83**

7 Conclusion **87**

List of Figures

1.1 Samples from GP priors with the RBF, Matern32, ArcCos and Quadratic kernels. The shaded region is 2 standard deviations around the mean (dotted line). . . . 5
1.2 A GP model depicted graphically. Right: the probabilistic graphical model. Left: the likelihood evaluation diagram. . . . 6
1.3 Sample from GP posteriors with likelihood variance 10^{-6} (top row) and 10^{-2} (bottom row). The kernels are as in Figure 1.1. . . . 8
1.4 The log densities for various likelihoods, demonstrating the smoothness. Integrating these log densities with quadrature under a Gaussian is likely to be reasonable accurate, even with a relatively small number of quadrature points. . . . 12

2.1 Two visulations of the two layer DGP prior, indicating the *statistical* perspective (left) and *selectional* perspective (right). The smooth curves are samples from the functions, whereas the crosses are points being propagated through the layers in the likelihoood. . . . 25
2.2 A two layer DGP model depicted graphically. Right: the probabilistic graphical model. Left: the likelihood evaluation diagram. . . . 25
2.3 As Figure 2.1 but with the identity mean function for the first layer. The chance of folds is reduced. . . . 26

2.4 Regression test log-likelihood results on benchmark datasets. Higher (to the right) is better. The sparse GP with the same number of inducing points is highlighted as a baseline. 30

3.1 The natural gradient (blue arrow) is correctly scaled and points in a better direction than the ordinary gradient (orange arrow). The contours show the lower bound to a GP with a Bernoulli likelihood and variational posterior with a single Gaussian inducing point. The path followed by taking very small steps in the ordinary gradient (orange curve) ascends the contours. The path taking small steps in the natural gradient (blue curve) is independent of parameterization, and does not follow the contours. . . . 36

3.2 The natural gradient is a superior *direction* in both parameterizations, and the best step size increases during optimization to $\gamma \approx 1$. Upper row: optimization methods, all with a line search for the step size. Lower row: the step sizes used at each iteration. 43

3.3 Stochastic optimization of the lower bound for fixed hyperparameters. The batch size is 256 and 5000 iterations are shown for five splits. 45

3.4 Joint optimization of the hyperparameters and the variational distribution. For natural gradients, a step of NGD on the variational parameters is alternated with a step of Adam on the hyperparameters. For Adam and GD the variational and hyperparameters are optimized together in a single objective. The batch size is 256 and 5000 iterations are shown for five splits. 45

3.5 Optimization of the NAVAL dataset ($N = 11$K, $D = 16$), with three different likelihoods. The ill-conditioning of the variational distributions renders the optimization using ordinary gradients extremely difficult, even given a large number of iterations and different values for the Adam learning rate. The batch size is 256 and 5000 iterations are shown for a single split. . . . 46

3.6 Optimization of hyperparameters and variational distributions for larger UCI datasets with a student-t likelihood. 47

3.7 Optimization of the hyperparameters and variational distribution for the MNIST data, with a Robust-max multiclass likelihood. We see that after the first few initial iterations NGD+Adam outperforms Adam alone. . . . 48

3.8 Optimization curves for a two layer DGP with fixed hyperparameters 49

4.1 The GP-CDE model. Left: the likelihood evaluation diagram. Right: the graphical model. 54

4.2 Three draws from the prior, with different variances for the latent variable. The outputs (orange) are projected onto the back pane. The surface is a single draw from the two dimensional GP prior. 54

4.3 The GP-CDE model, with probabilistic input projections. Left: the likelihood evaluation diagram. Right: the graphical model. 57

4.4 Conditional densities (displayed as heat-maps: yellow means higher probability) of drop-off locations conditioned on the pick-up location (red cross). 59

4.5 Sample images for 4-shot learning. Left column is a true (unseen) image, remaining columns are samples from the posterior conditioned on the same label. 61

4.6 Test log-likelihood of the GP, the optimal GP-CDE, the amortized GP-CDE, the CVAE, and a Linear model on 10 UCI datasets. Hihgher is better. 61

4.7 The mean of the GP mapping conditioned on the 1D latent variable, for a dataset of 100 '1's. Top: the analytic solution of Titsias and Lawrence [2010a]. Middle: our natural gradient approach, with a step size of 0.1. Bottom: using the Adam optimizer to optimize the variational parameters $\mathbf{m}$ and $\mathbf{S}$. We see that the ordinary gradient approach is prone to getting 'stuck' in poor local optima, due to the difficulty of optimization. 63

4.8 The training objective for the dataset of 100 '1's. The three plots show the same three curves with different ranges on the y-axis to highlight the similarities and differences. We can see that natural gradients provide a striking improvement. See Fig. 4.7 for the samples at the end of training 64

5.1 Posterior density from a two-layer model, illustrating non-Gaussian marginals and non-smooth input dependence. 66

5.2 The DGP with latent variables in two configurations: GP-LV-GP (left) and LV-GP-GP (right). The graphical model (centre) is the same for each. 68

5.3 A single sample from the prior for 4 illustrative models. The GP and GP-GP model have no latent variables. Their samples appear as deterministic functions. 68

5.4 Posteriors for 1D synthetic data. The models without latent variables cannot capture the bimodality or heteroscedasticity. See Figure 5.1 for the LV-GP-GP-GP model. 73

5.5 Samples from the predictive distribution at test points (orange) together with the actual value (blue) in the PCA projection to 2D. . . . 76

5.6 Posterior marginals, re-scaled to zero mean and unit standard deviation, with the Gaussian density marked as a dotted curve. Colors are according to the principal component of the inputs. These are the four datasets highlighted in the main text for being prominent examples benefiting from latent variables. In these cases the additional depth of model leads to more non-Gaussian marginals. 81

5.7 As Figure 5.6, but for the four datasets highlighted in the main text for benefiting from both latent variables and depth. Note that, unlike in Figure 5.6, for the 'pol' and 'bike' data the deeper models actually have *simpler* marginals. This is because the deep model has more complexity in the mapping, so can more closely fit the data, whereas the simple model must explain the data with a complex noise distribution. 82

6.1 Figure from Hebbal et al. [2019], showing the variables to be optimized using the DGP as a surrogate function within the Bayesian Optimization framework 85

6.2 Figures from Hebbal et al. [2019], showing a sectional view of the change in velocity according to the diameters of the nozzle of the rocket booster in Figure 6.1, demonstrating the non-stationary behaviours of the functions involved in the optimization problem. 85

6.3 Figure from Koriyama and Kobayashi [2019], showing the quantitative improvements of the DGP model in speech synthesis. The mel-cepstrum distance is a metric on the representation of the short-term power spectrum of a sound (lower better is better). 86

6.4 Left: the results from Svendsen et al. [2018]. Right: illustrative data used in Svendsen et al. [2018], taken from Aires et al. [2002]. Clearly such data is poorly modelled by a single layer GP due to the heteroscedastic noise and non-stationarity, but is more appropriate for a deep model. 86

List of Tables

2.1 Results on Rectangles-Images dataset ($N = 12000$, $D = 784$) 32

2.2 Regression test RMSE results for large datasets 32

2.3 Regression test log likelihood results for large datasets 32

2.4 Typical computation time in seconds for a single gradient step. All models have 100 inducing points, except SGP 500, which has 500 . . 33

4.1 NLPP for Manhattan data (lower is better). The models are trained on different dataset sizes. 60

4.2 Log-likelihoods of the CVAE and GP-CDE models. N is the number of images per class. Higher test log-likelihood is better. 63

5.1 Average test log-likelihood results for five splits (standard error). **Bold** font indicates overlapping error bars with the best performing method. *Italic* indicates a significantly non-Gaussian posterior, computed by the Shapiro Wilk test (see Table 5.3). Colours indicate datasets demonstrating prominent examples of complex **marginals***, **mappings**†, or **both**‡. 75

5.2 Test log likelihood values over 5 splits (standard errors), including the CVAE models and models with Stochastic Gradient HMC Havasi et al. [2018b]. Results less than -1000 are reported as $-\infty$. The numbers after the CVAE models are the number of hidden units. The SGHMC results were run with identical models as the corresponding VI methods, Hyperparameter optimization was performed using a random sample from the last 100 iterations, as described in Havasi et al. [2018b]. Posterior sample were take with 2000 samples, with a thinning factor of 5 (i.e. a chain of length 10000). We included also a single layer GP using the SGHMC approach for comparison. The discrepancy between the variational approach and the SGHMC is attributable to several factors (note that the variational approach is optimal in this case): different batch sizes (512 in this work compared to 10000 in Havasi et al. [2018b]); different hyperparameter initializations (for example, a much longer initial lengthscales is used in Havasi et al. [2018b]); different learning rate schedules (decaying learning rate in this work compared to fixed in Havasi et al. [2018b]); the use of natural gradients in this work for the final layer, as opposed to optimizing with Adam optimizer, as done in Havasi et al. [2018b], Salimbeni and Deisenroth [2017b]; insufficient mixing in the posterior chain. 79

5.3 Median Shapiro Wilk test statistic for the test points. Blanks have values of one (perfectly Gaussian). Smaller values indicate more evidence for non-Gaussian marginal distributions. 80

Chapter 1

Introduction

This thesis is concerned with building models from compositions of Gaussian processes. There are several motivations for this endeavour. The first is that Gaussian processes are limited in ways that cannot be overcome within the single layer framework. Combining multiple Gaussian processes in a hierarchy through function composition is one of many potential alternatives to overcome some of these limitations. The limitations of (single layer) Gaussian processes, and approaches to overcome them, are discussed in Section 1.5 after the essential background material has been introduced.

A second motivation can be found in the striking successes and also striking failures of deep neural networks. Prominent successes include image classification [Krizhevsky et al., 2012], natural language translation [Wu et al., 2016b], audio synthesis [Van Den Oord et al., 2016] and reinforcement learning [Silver et al., 2016]. Prominent failures include ill-calibrated uncertainty [Kuleshov et al., 2018], a propensity to misclassify out-of-sample examples with high confidence [Nguyen et al., 2015], and fragility to adversarial attacks [Jia and Liang, 2017]. Additional drawbacks include the reliance on huge data and huge compute [Lucic et al., 2018], lack of interpretability [Zhang and Zhu, 2018], a lack of mechanisms to incorporate known prior knowledge such as the laws of physics [Marcus, 2018], and the reliance on a large number of poorly understood techniques to ensure good performance [see, for example Sutskever et al., 2013, Ioffe and Szegedy, 2015]. Both the successes and failures of deep neural networks are inspirations for investigating deep Gaussian processes.

Motivation: the successes of deep learning

The success of deep neural network models suggest that compositions of simple functions are effective for solving real world problems. The reasons for this are not well understood [Zhang et al., 2017]. One argument posits that observed world may have inherently compositional structure most readily captured by a hierarchical model [Lin et al., 2017]. Another suggests that hierarchical representations of information are essential for reasoning [Hinton, 2007], appealing to the biological brain for analogy. For whatever the reason, the empirical success of deep networks suggests that different approaches to function composition might be worth pursuing. A deep Gaussian process is one way to compose functions within the Bayesian framework.

Motivation: the failures of deep learning

The failures of deep networks also provide motivation for considering a deep Gaussian process. The Bayesian approach has been shown to provide a solution to some of the problems of neural networks [MacKay, 1991, Neal, 1996], for example by guarding against overfitting and providing a meaningful way to perform model comparison. The Bayesian approach places priors over the weights of a neural network, rather than the using point estimates. It is not clear why these priors should be chosen, however, as a prior on weights gives rise to a prior over mappings that is difficult to interpret, even for a single layer. A deep Gaussian process provides an alternative mechanism for building deep Bayesian models. As a Gaussian process is a prior directly over functions, rather than indirectly over weights, it is possible to reason directly over the mappings at each layer and incorporate interpretable prior information such as smoothness and periodicity. A deep Gaussian process can be 'tuned' with a small number of hyperparameters, varying from a simple model to a complex one.

Motivation: the intersection of black-boxes with model-based approaches

Breiman et al. [2001] famously contrasted 'machine learning' with 'statistics'. In this characterization, statistics seeks to model the mechanism of the world directly, whereas machine learning is only concerned with prediction. A more nuanced view is certainly possible [see, for example, the published comments to Breiman et al., 2001]: machine learning models can be constructed to take advantage of known properties of the world, for example convolutional layers are motivated by physical considerations of translation invariance; statistical methods can be designed to allow flexibility so their assumptions are not too restrictive, for example non-parametric models such as Gaussian processes.

The Gaussian process framework allows for the incorporation of prior knowledge while maintaining a flexible model. For example, Solin et al. [2018] builds a covariance function for a magnetic field that explicitly obeys Maxwell's equations, but allows

for arbitrarily complex form of the underlying potential. This approach is both extremely restrictive (it requires that Maxwell's equations are exactly true) and extremely flexible (the underlying field potential could take any form).

Another way of expressing the argument of Breiman et al. [2001] is through black-boxes: machine learning is characterized by building 'black-box' models, the insides of which are not relevant: the only thing that matters is how well the outputs match observations. A Gaussian process can fit any function given sufficient data for many standard choices of kernel, so in this sense a Gaussian process does not impose rigid assumptions and can be used as a black-box method in the spirit of machine learning. While Gaussian processes can be made very flexible, they still carry assumptions that can be excessively restrictive, however. It will be argued at the end of this chapter there are some phenomena that can never be modelled by a Gaussian process.

There are many ways to obtain models that do not have the restrictions imposed by a simple Gaussian process model, but typically these models are largely black-box and it is not possible to incorporate meaningful prior assumptions. For example, a Bayesian neural network is certainly a very flexible model, but it is not straightforward to incorporate prior assumptions such as Maxwell's equations. A deep Gaussian process can provide a middle ground between the black-box and model-based alternatives. A deep Gaussian process constructed as described in Chapter 2 has the useful property of containing the single layer Gaussian process as a special case. With a very small number of hyperparameters it is possible to smoothly transform from a single layer Gaussian process (which can be interpretable and can include strong prior assumptions) to a model which is essentially black-box.

Motivation: a deep Bayesian model with semi-tractable inference

Inference in deep Gaussian process models is intractable. Fortunately, methods in approximate inference for the single layer Gaussian process can be adapted to the deep case. This adaptation is the subject of much of this thesis. A key property of the variational approach is the separation between representing a *function* and representing its *values* at the data points. It is this property that is exploited in Chapter 2 to achieve a method of approximate inference that is both effective and scalable.

Recent advances in Bayesian inference have proposed fully automatic inference engines that remove much of the cumbersome details traditionally associated with Bayesian methods [Kucukelbir et al., 2017]. These approaches generally make mean-field assumptions and cannot exploit model specific algebraic details, however. The inference proposed in this thesis is highly specific to the deep Gaussian process model, but also captures crucial structure that would be missing by a mean field approach. Very recent work [Tran et al., 2018] has adapted the approaches presented in this thesis to a probabilistic programming paradigm. This approach is very exciting as it exploits the model structure while remaining easy to use and easy to combine with other models.

Chapter summary

The remainder of this chapter introduces the foundational material on Gaussian processes and variational inference (Section 1.2), including the extensions to multi-output models and a variety of approaches to kernel construction. The remainder of the chapter is devoted to discussing the limitations of Gaussian processes, and the attempts to overcome these limitations within the single layer framework.

1.1 Gaussian processes

A stochastic function assigns elements of an index set $\mathcal{X}$ to a random variable $f(x)$. The complete set of variables (the 'function') is notated as $f := \{f(x)\}_{x\in\mathcal{X}}$. A stochastic process is defined by its finite marginal distributions $p(\{f(x_1), \ldots, f(x_N)\})$. Note that 'marginal' here means a subset of variables, and not the product of univariate distributions. The only criterion for the process be well defined is that the marginals are consistent [Kolmogorov, 1933] under permutation and marginalisation.

A Gaussian process prior is a stochastic process where the finite marginals subset are jointly Gaussian, with mean and covariance that depend on the input locations. As the mean and variance are therefore deterministic functions of the input. The mean function is a function from the index set $\mathcal{X}$ to the reals, and the covariance function is a function from $\mathcal{X} \times \mathcal{X}$ to the reals. For example, using m and k for the mean and covariance functions respectively, the distribution for two variables $f(x_1)$ and $f(x_2)$ is:

$$\begin{pmatrix} f(x_1) \\ f(x_2) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(x_1) \\ m(x_2) \end{pmatrix}, \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) \\ k(x_2,x_1) & k(x_2,x_2) \end{pmatrix}\right) \tag{1.1}$$

The mean function can be any function, but the covariance function must be positive definite, that is $\sum_{ji} v_i k(x_i, x_j) v_j \geq 0$ for all v_i and x_i. To complete the specification of the prior we need to specify the mean and covariance functions. Any positive definite covariance function can be chosen for k. Four well known covariance functions are

Kernel	$k(x, x')$
RBF	$\exp\left(-r^2\right)$
Matern$_{\frac{3}{2}}$	$(1 + \sqrt{3}r)\exp\left(-r\right)$
ArcCosine	$\sin\theta + (\pi - \theta)\cos\theta$
Quadratic	$(x \cdot x' + 1)^2$

where $r^2 = \sum_d (x_d - x'_d)^2$ and d is the input dimension, and $\cos\theta = \frac{x \cdot x'}{|x||x'|}$. The above covariance functions are generally defined with scaling constants, but these have been omitted for the present. Prior samples from Gaussian processes with these covariance functions are shown in Figure 1.1. Five samples at 500 points uniformly spaced in the interval $[-3, 3]$ are plotted together with the mean (dotted line), which in each case is zero, and the 2 standard deviations error bars. The four covariance functions give rise to very different samples, and very different marginal variances.

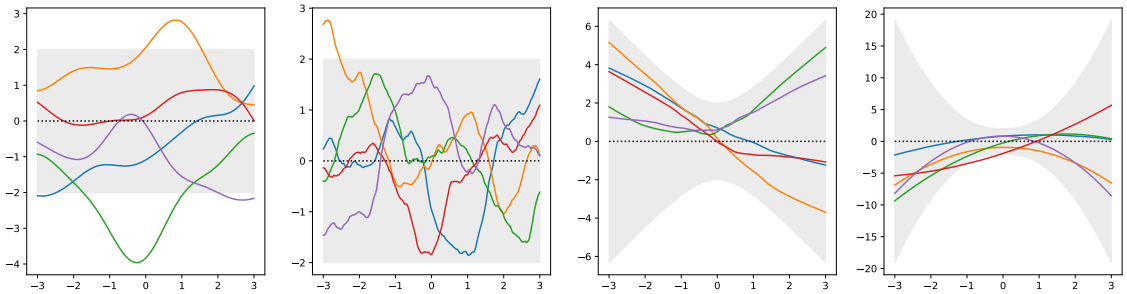


Figure 1.1: Samples from GP priors with the RBF, Matern32, ArcCos and Quadratic kernels. The shaded region is 2 standard deviations around the mean (dotted line).

Models for data can be constructed with Gaussian process priors. In the supervised setting the data are given as N observations y_n at corresponding input locations x_n. The shorthand $y = \{y_n\}_{n=1}^N$ and similarly for x is used to lighten the notation. A common approach is to assume an observation model of the form $p(y|f,x) = \prod_{n=1}^N p(y_n|f(x_n))$. That is, the observations are conditionally independent given the latent function value at the corresponding input location. In this case the joint distribution[1] is given by

$$p(f,y) = p(f)\prod_{n=1}^{N} p(y_n|f(x_n))\,. \tag{1.2}$$

A graphical model is shown in Figure 1.2, indicating the prior *statistical* relationship between the variables in our model. Note that the prior for f is jointly over the potentially infinite collection of variables, so we notate f as a single node in the graphical model (with an infinity symbol to emphasize that the collection may be infinite). It is useful to notate how variables from the full collection are selected and associated with other variables, so we introduce a graphical representation which we call a *likelihood evaluation diagram*, shown in Fig. 1.2 alongside the graphical model. The purpose of this notation is to make clear how the Gaussian process is indexed in the likelihood. This information cannot be captured by the graphical model, so the two diagram as complementary: the evaluation diagram indicates the *selectional* relationship between variables, whereas the graphical model indicates the *statistical* relationships between variables. It might seem tempting to include only the finite variable associated with the data in the graphical model. This is problematic, however, and would lead to significant confusion in the multi-layer case. To see the problem, consider the case where the index set has only two elements, for example $\{0,1\}$. In this situation there are exactly two variables in the model: $f(0)$ and $f(1)$. If there are N observations, then defining $f_n = f(x_n)$ and drawing each f_n in a graphical model is not correct as there are only two variables, but N nodes in the graphical model. The statistical dependency between the f_n is not straightforward: if $f(0)$ and $f(1)$ are independent then $\mathrm{cov}(f_1, f_2)$ is either zero or

[1]For the rest of this section we will assume that $\mathcal{X}$ is finite so we can write $p(f)$. The arguments can be extended to the infinite case, but more mathematical machinery is required. For a detailed description see Matthews et al. [2016b]

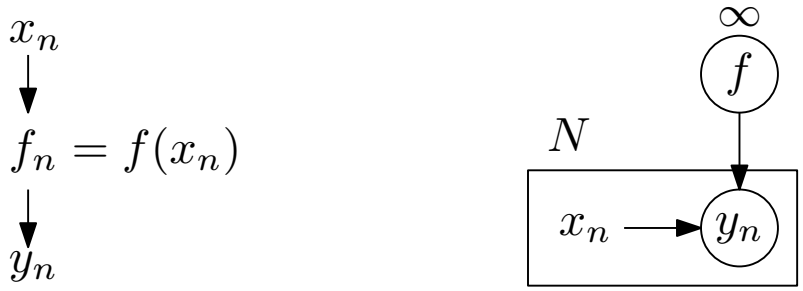


Figure 1.2: A GP model depicted graphically. Right: the probabilistic graphical model. Left: the likelihood evaluation diagram.

one, depending on whether or not x_1 has the same value as x_2.

The previous paragraph argued against considering only $f_n = f(x_n)$ in the definition of the model, but it is nonetheless helpful to distinguish those associated with data points and those that are not. The notation $\mathbf{f}$ is used for the function values at the input locations $[\mathbf{f}]_i = \{f(x_n)\}_{n=1}^N$, and f_* for the remaining function values at all other locations $f_* = \{f(\mathbf{x})\}_{\mathbf{x}\in\mathcal{X}\setminus\mathcal{D}}$. Abbreviating the likelihood terms as $p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^N p(y_n|f(x_n))$, the joint distribution (1.2) can then be written

$$p(f, y) = \underbrace{p(f_*|\mathbf{f})}_{\text{projection}} \underbrace{p(\mathbf{f})p(\mathbf{y}|\mathbf{f})}_{\text{data term}} . \tag{1.3}$$

Crucially, the expression in (1.3) is identical to (1.2) except the prior has been factorized into a certain partition of the variables. Equation (1.3) highlights the *data term*, $p(\mathbf{f})p(\mathbf{y}|\mathbf{f})$, concerning the N function values at the observations, and the *projection term*, $p(f_*|\mathbf{f})$, concerning all the remaining function values. The projection term is so named as it projects the data term to the remaining variables that interacted with the data only through the prior, and not through the likelihood. By considering any finite subset of f_* and using standard result for Gaussian conditionals, the projection term can be shown to take the form of another Gaussian process, with mean function $\mu_{\mathbf{f}}$ and covariance function Σ, where

$$\mu_{\mathbf{f}}(x) = m(x) + \mathbf{k}(x)^\top \mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}) \tag{1.4}$$

$$\Sigma(x, x') = k(x, x') - \mathbf{k}(x)^\top \mathbf{K}^{-1}\mathbf{k}(x') , \tag{1.5}$$

with $[\mathbf{k}(x)]_i = k(x_i, x)$, $[\mathbf{K}]_{ij} = k(x_i, x_j)$ and $[\mathbf{m}]_i = m(x_i)$.

In the special case that the likelihood is Gaussian, $p(y|f(x)) = \mathcal{N}(y|f(x), \sigma^2)$, the data term can be computed analytically using the standard result for the product of two Gaussians [2]. The result is proportional to an N-dimensional Gaussian with

[2] That is, $\mathcal{N}(a|\mu_a, \Sigma_a)\mathcal{N}(Ba|\mu_b, \Sigma_b) = \mathcal{N}(a|\Lambda\left(\Sigma_a^{-1}\mu_a + B^\top\Sigma_b^{-1}B\mu_b\right), \Lambda)\mathcal{N}(\mu_a|B\mu_b, \Sigma_a + B^\top\Sigma_b B^\top)$ where, $\Lambda^{-1} = \Sigma_a^{-1} + B^\top\Sigma_b^{-1}B$. Note that the $\mathcal{N}(x|a, b)$ notation defines a function of x, where as $\mathcal{N}(a, b)$ defines a random variable

mean $\bar{\mathbf{m}}$ and covariance $\bar{\mathbf{S}}$, where

$$\bar{\mathbf{m}} = \bar{\mathbf{S}}(\mathbf{K}^{-1}\mathbf{m} + \sigma^{-2}\mathbf{y}) \tag{1.6}$$
$$\bar{\mathbf{S}} = (\mathbf{K}^{-1} + \sigma^{-2}\mathbf{I})^{-1}\,, \tag{1.7}$$

where the constant of proportionality is given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \mathbf{K} + \sigma^2\mathbf{I})\,. \tag{1.8}$$

The above quantity is the known as the marginal likelihood, or evidence. Section 1.4 describes approaches that use this quantity to learn a suitable kernel function.

Noting that the mean of the projection term (1.4) is linear in $\mathbf{f}$, standard results for linear Gaussian systems can be used to marginalize $\mathbf{f}$. The result is jointly Gaussian for all f, i.e., another Gaussian process. The mean and covariance functions of this process are given by $\bar{\mu}$ and $\bar{\Sigma}$, where,

$$\bar{\mu}(x) = m(x) + \mathbf{k}(x)^\top \mathbf{K}^{-1}(\bar{\mathbf{m}} - \mathbf{m}) \tag{1.9}$$
$$\bar{\Sigma}(x, x') = k(x, x') - \mathbf{k}(x)^\top \mathbf{K}^{-1}\mathbf{k}(x') + \mathbf{k}(x)^\top \mathbf{K}^{-1}\bar{\mathbf{S}}\mathbf{K}^{-1}\mathbf{k}(x')\,. \tag{1.10}$$

Substituting (1.6) and (1.7) into (1.9) and (1.10) gives the familiar expressions,

$$\bar{\mu}(x) = m(x) + \mathbf{k}(x)^\top (\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\bar{\mathbf{y}} - \mathbf{m}) \tag{1.11}$$
$$\bar{\Sigma}(x, x') = k(x, x') - \mathbf{k}(x)^\top (\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(x')\,. \tag{1.12}$$

Figure 1.3 shows samples from the posterior processes with mean and covariance functions given by (1.11) and (1.12)) for each of the four examples from Figure 1.1 with three observations and a Gaussian likelihood. The top row shows posteriors with a very small likelihood variance of $\sigma^2 = 10^{-6}$, and the bottom row with value of $\sigma^2 = 10^{-2}$. The smaller likelihood variance forces the posterior samples to pass very close to the data.

1.2 Variational Inference

Inference in Gaussian process models is intractable if the number of data points is large or if the likelihood is non-Gaussian. Variational inference provides a way to deal with both these difficulties simultaneously. The key idea is to posit an approximate posterior over full set of (infinitely many) variables, and then minimize the Kullback-Leibler (KL) divergence from the approximate posterior to the true posterior. The KL divergence is defined as

$$\mathrm{KL}(q(f)||p(f)) = \mathbb{E}_{q(f)} \log \frac{q(f)}{p(f)} \tag{1.13}$$

The approximate posterior has a special structure that allows the infinite number of variables to be captured with finite computation.

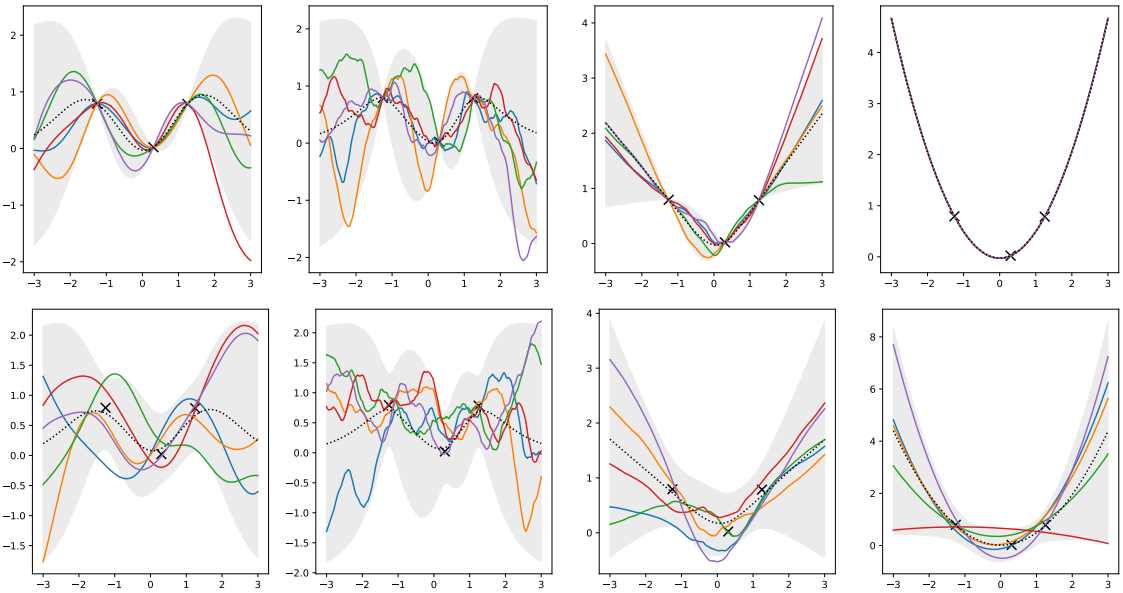


Figure 1.3: Sample from GP posteriors with likelihood variance 10^{-6} (top row) and 10^{-2} (bottom row). The kernels are as in Figure 1.1.

The approximate posterior will be a product of two parts. The first is a special Gaussian process which matches the prior Gaussian process at some fixed *inducing locations* that we are free to choose. It is this matching with the prior that allows infinite variables to be modeled with finite computation. The second part is a distribution over variables at the inducing locations (the 'inducing points'). This distribution is further assumed to be Gaussian, which turns out to be optimal in the case of Gaussian likelihood.

A key advantage of the variational posterior is that the marginals are Gaussian, with mean and variance that are linear and quadratic functions of the kernel evaluated at the inputs. This is in contrast with the exact posterior marginals, which depend non-linearly on all the inputs. This decoupling allows the use of minibatches, and will be central later for inference in deep Gaussian process models.

Inference with Gaussian process models presents two sources of intractability: analytical and computational. The analytic intractability arises whenever the likelihood is not Gaussian (so $p(f)$ is not conjugate to $p(y|f)$), and the computational intractability comes from the manipulation of matrices of size $N \times N$. Variational inference is a general approach for approximating intractable posteriors which can overcome both these difficulties simultaneously [Hensman et al., 2015].

Variational inference seeks an approximate posterior $q(f)$ that is made as close as possible to the true posterior $p(f|y)$, where the closeness is measured by the Kullback-Leibler divergence $\mathrm{KL}(q(f)||p(f|y))$. The basis of variational inference is

the identity

$$\log p(y) = \underbrace{\mathbb{E}_{q(f)} \log \frac{p(y|f)p(f)}{q(f)}}_{\mathcal{L}_q} + \text{KL}(q(f)||p(f|y)) \,. \tag{1.14}$$

The left hand side of (1.14) is independent of $q(f)$, so minimizing the KL divergence from the approximate posterior to the true posterior is equivalent to maximizing $\mathcal{L}_q$. Since KL is positive and equal to zero only when the two distributions are the same, it follows that $\mathcal{L}_q$ is a lower bound on $\log p(y)$. The maximization of $\mathcal{L}_q$ therefore serves a dual purpose. The motivation is to find the closest approximating distribution to the true posterior, where the closeness is measured in KL divergence, but a consequence of (1.14) is that the resulting objective is also a lower bound on the model evidence. The lower bound can also be written as

$$\mathcal{L}_q = \mathbb{E}_{q(f)} \log p(y|f) - \text{KL}(q(f)||p(f)) \tag{1.15}$$

A subtlety of variational inference in non-parametric models is that we need to place a variational distribution over the entire collection of variables f, not just the variables associated with the data. Throughout this thesis the variational distribution will be another Gaussian process with a special form, The form, first proposed by Titsias [2009], begins with a subset of the inputs $\mathcal{V} = \{\tilde{x}_m\}_{m=1}^M$, which will be referred to as *inducing inputs*. The Gaussian process variables f can be partitioned into two disjoint subjects: the function values at the inducing points $\{f(\tilde{x}_m)\}_{m=1}^M$, which can be stacked to a vector $\tilde{\mathbf{f}}$, and the remaining variables $f_* = \{f(x)\}_{x \in \mathcal{X} \setminus \mathcal{V}}$ (note this is a new definition of f_* compared to (1.3), although we excuse the inconsistency by highlighting the intent to abuse notation and use f_* to refer to all remaining variables implied by context). The variational distribution is given the form

$$q(f) = p(f_*|\tilde{\mathbf{f}})q(\tilde{\mathbf{f}}) \,, \tag{1.16}$$

where $p(f_*|\tilde{\mathbf{f}})$ is the prior conditional. This term is exactly the projection term from (1.3), but with the conditioning on the inducing points rather than the function values at the data, i.e. a Gaussian process with mean and covariance given by the expressions in (1.4) and (1.5) but with all data terms replaced with the equivalent inducing point terms. That is, $p(f_*|\tilde{\mathbf{f}})$ is a Gaussian process with mean and covariance functions given by $\mu_{\tilde{\mathbf{f}}}$ and $\Sigma_{\tilde{\mathbf{f}}}$, which are exactly the functions in (1.4) and (1.5), but with different inputs. The forms are

$$\mu_{\tilde{\mathbf{f}}}(x) = m(x) + \tilde{\mathbf{k}}(x)^\top \tilde{\mathbf{K}}^{-1}(\tilde{\mathbf{f}} - m(\tilde{\mathbf{X}})) \tag{1.17}$$

$$\Sigma_{\tilde{\mathbf{f}}}(x, x') = k(x, x') - \tilde{\mathbf{k}}(x)^\top \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{k}}(x') \,, \tag{1.18}$$

with $[\tilde{\mathbf{k}}(x)]_i = k(\tilde{x}_i, x)$ and $[\tilde{\mathbf{K}}]_{ij} = k(\tilde{x}_i, \tilde{x}_j)$.

The prior can be expressed using the same partition of variables as in (1.16),

$$p(f) = p(f_*|\tilde{\mathbf{f}})p(\tilde{\mathbf{f}}) \,. \tag{1.19}$$

This decomposition is possible because both the prior and variational distribution are over the same set of (infinitely many) variables. Considering only the variables associated with the data and augmenting with inducing inputs is not valid in general [see Matthews, 2017, for details]. Substituting expressions (1.16) and (1.19) into (1.15) the evidence lower bound

$$\mathcal{L} = \mathbb{E}_{q(f)} \log \prod_{n=1}^{N} p(y_n|f(x_n)) - \mathrm{KL}(q(f)||p(f))\,. \tag{1.20}$$

Both terms (1.20) simplify. The KL term simplifies due to the cancellation of the prior conditional:

$$\begin{aligned}
\mathrm{KL}(q(f)||p(f)) &= \mathbf{E}_{q(f)} \log \frac{p(f)}{q(f)} && (1.21)\\
&= \mathbf{E}_{p(f_*|\tilde{\mathbf{f}})q(\tilde{\mathbf{f}})} \log \frac{p(f_*|\tilde{\mathbf{f}})p(\tilde{\mathbf{f}})}{p(f_*|\tilde{\mathbf{f}})q(\tilde{\mathbf{f}})} && (1.22)\\
&= \mathbf{E}_{p(f_*|\tilde{\mathbf{f}})q(\tilde{\mathbf{f}})} \log \frac{p(\tilde{\mathbf{f}})}{q(\tilde{\mathbf{f}})} && (1.23)\\
&= \mathbf{E}_{q(\tilde{\mathbf{f}})} \log \frac{p(\tilde{\mathbf{f}})}{q(\tilde{\mathbf{f}})} && (1.24)
\end{aligned}$$

The above derivation is based on the existence of the Gaussian process density, which is not defined for infinite index set. The KL divergence can be defined for infinitely many variables using the RadonâĂŞNikodym derivative, and this approach is developed for the Gaussian process case in Matthews [2017]. The key idea is to define the variational posterior as a ratio with the prior, rather than through a density. For the purposes of this thesis the variational distribution will be defined as if it had a density, but with awareness that the construction is only valid if it corresponds to a well-defined KL divergence with the prior. The construction in (1.16) does yield a valid KL divergence because the variational distribution only differs from the prior at finitely many points.

The first term in (1.20) simplifies as follows

$$\begin{aligned}
\mathbb{E}_{q(f(x_n))} \log \prod_{n=1}^{N} p(y_n|f(x_n)) &= \mathbb{E}_{q(f)} \sum_{n=1}^{N} \log p(y_n|f(x_n)) && (1.25)\\
&= \sum_{n=1}^{N} \mathbb{E}_{q(f(x_n))} \log p(y_n|f(x_n)) && (1.26)
\end{aligned}$$

Putting these two terms together, the bound is

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(x_n))} \log p(y_n|f(x_n)) - \mathrm{KL}[q(\tilde{\mathbf{f}})||p(\tilde{\mathbf{f}})]\,. \tag{1.27}$$

There are two important features of this bound, which is known as the Evidence

Lower Bound, or ELBO. Firstly, it involves only finite objects due to the cancellation of the $p(f_*|\tilde{\mathbf{f}})$ terms. Secondly, to calculate the bound we need only the univariate marginals of the variational posterior at the data locations, i.e. we need only calculate $q(f(x))$.

It is possible to optimize for $q(\tilde{\mathbf{f}})$ directly [Matthews et al., 2016b], or alternatively it can be assumed Gaussian $q(\tilde{\mathbf{f}}) = \mathcal{N}(\tilde{\mathbf{m}}, \tilde{\mathbf{S}})$. As $\mu_{\tilde{\mathbf{f}}}$ is linear in $\tilde{\mathbf{f}}$, standard results for linear Gaussian systems show that $q(f)$ is a Gaussian process with mean and covariance functions $\mu_{\tilde{\mathbf{m}}}$ and $\Sigma_{\tilde{\mathbf{S}}}$, where

$$\mu_{\tilde{\mathbf{m}}}(x) = m(x) + \tilde{\mathbf{k}}(x)^\top \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{m}} \tag{1.28}$$

$$\Sigma_{\tilde{\mathbf{S}}}(x, x') = k(x, x') + \tilde{\mathbf{k}}(x)^\top \tilde{\mathbf{K}}^{-1} (\tilde{\mathbf{S}} - \tilde{\mathbf{K}}) \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{k}}(x') \,. \tag{1.29}$$

These functions are identical to those in the exact posterior (1.9) and (1.10), but the inputs are $\tilde{\mathbf{X}}$ rather than the data locations $\mathbf{X}$, and $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$ are variational parameters, rather than functions of the data (also, the mean function at the inducing points has been absorbed into $\tilde{\mathbf{m}}$). In contrast to the $\mathcal{O}(N^3)$ complexity for evaluating the exact posterior, these function have complexity $\mathcal{O}(M^3)$ to evaluate, providing a computation saving if $M \ll N$.

The likelihood term can be computed analytically in the case of the Gaussian likelihood as $\log p(y_n|f(x_n))$ is a quadratic form in $f(x_n)$. The expectation under $q(f(x_n))$ is tractable since $q(f(x_n))$ is Gaussian with mean $\mu_{\tilde{\mathbf{m}}}(x_n)$ and variance $\Sigma_{\tilde{\mathbf{S}}}(x_n, x_n)$. Since $\mu_{\tilde{\mathbf{m}}}(x_n)$ is linear in $\tilde{\mathbf{m}}$ the expectation is quadratic in $\tilde{\mathbf{m}}$ and linear in $\tilde{\mathbf{S}}$. The optimal solution can be found for $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$, as first shown in Titsias [2009]. To evaluate the solution has complexity $\mathcal{O}(NM^2 + M^3)$, which can be parallelized over N [Gal et al., 2014].

In the non-Gaussian likelihood case the log-likelihood expectations in (1.27) can be approximated with quadrature, or are available in closed form for Poisson and Exponential likelihood (when used with the exponential link function). Note that the integral is of the *log*-likelihood, and not the likelihood density itself, so even 20 quadrature points might feasibly give a good approximation. This can be seen in plots of the Student-t, Bernoulli, gamma, Beta and Ordinal likelihoods shown in Figure 1.4. The link function is the exponential for the Beta and Gamma likelihoods, and sigmoid for the Bernoulli. The Beta is modelled via the mean with the sigmoid link function and unit scale. The ordinal has switch points at 0 and 1. In all cases the log density is very smooth and easy to integrate numerically with high accuracy. The integral could also be approximated by Monte Carlo [Gal et al., 2015, Krauth et al., 2016b].

The variational objective is a function of the parameters $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$, the inducing locations $\{\tilde{x}\}_{m=1}^M$ and also the model parameters. The bound should be optimized with respect to the inducing locations, as these are parameters of the variational distribution $q(f)$, and not the model. It is common to further optimize the bound with respect to the model hyperparameters, analogously to optimizing the true marginal likelihood with respect to model parameters. This approach should be treated with caution, however, as the slack in the bound is not uniform over the hyperparameters. For example, a short lengthscale may lead to a poor inducing point

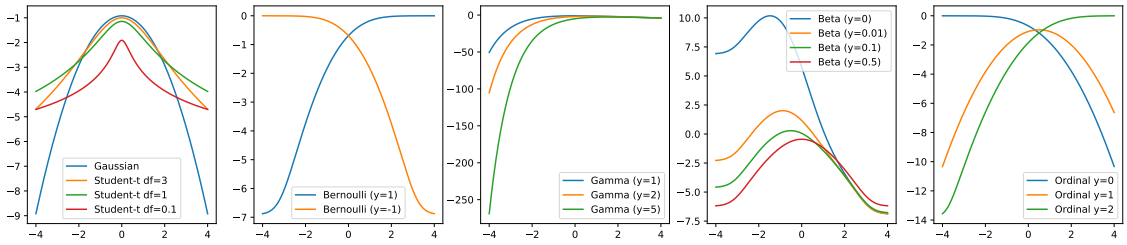


Figure 1.4: The log densities for various likelihoods, demonstrating the smoothness. Integrating these log densities with quadrature under a Gaussian is likely to be reasonable accurate, even with a relatively small number of quadrature points.

approximation if there are too few inducing points, even if the short lengthscale is the best model. Fortunately the bias introduced will favour models with true posteriors that are well approximated by the variational distribution, which is likely to be a desirable property. This effect is explored in further detail in Bauer et al. [2016], where an alternative approximation based on expectation propagation is shown to find inferior models, even when the inducing points are sufficient to learn a close approximation of the true model.

To calculate the bound over the full data has complexity $\mathcal{O}(NM^2 + M^3)$ due to the matrix multiplication for each data point. As the ELBO consists of terms over the data and the KL term, it is possible to obtain an unbiased stochastic estimate by sub-sampling the data terms and rescaling. This might be essential for practical considerations if the dataset is extremely large [Hensman et al., 2013]. It is also possible to apply the natural gradient [Amari, 1998] for this optimization. This possibility is explored in detail in Chapter 3.

In the case of the Gaussian likelihood is possible to refrain from making the Gaussian assumption for $q(\tilde{\mathbf{f}})$ and show that the optimal form is in fact Gaussian [Titsias, 2009]. In this case the optimal $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$ parameters can be computed directly. One reason not to do this, however, is that it precludes the use of minibatching. In this case the true posterior is recovered by placing the inducing points at the data, though this incurs the cubic complexity of the exact posterior.

1.3 Multiple outputs

The GP models so far have been scalar valued. To extend the framework to vector valued outputs (known as a 'multi-output' GP [Alvarez et al., 2012]) it is possible to consider the outputs in a flattened representation using the dimension index as an additional covariate. That is, if the outputs are $f^d(x)$ then $\mathrm{cov}(f^d(x), f^{d'}(x')) = k([x, d], [x', d'])$, where k is a covariance function on the augmented space and $[\,\cdot\,,\,\cdot\,]$ denotes concatenation. Exact inference follows before, except now the dataset is made D times larger as each observation contributes D points in the flattened representation, where D is the number of outputs. Computation in the conjugate model scales as $(ND)^3$ complexity and $(ND)^2$ memory.

There are some important special cases that simplify the relationship between outputs. First is the case of independent outputs, $k([x,d],[x',d']) = \delta_{dd'}k_d(x,x')$. In this case exact inference for a Gaussian likelihood is exactly as before just with D independent calculations for each output y^d. Computation in the conjugate model then scales as N^3D complexity and N^2D memory. Variational inference is a straightforward extension of before as each output can be treated independently. The variational parameters $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$ and $\{z_m\}_{m=1}^M$ should be distinct for each output ($\tilde{\mathbf{m}}^d$, $\tilde{\mathbf{S}}^d$ and $\{z_m^d\}_{m=1}^M$). A further simplification can be made if the covariance is the same for each output $k_d(x,x') = k(x,x')$. In this case it is possible to reuse computation if the inducing points are also shared.

Another simple case for multi-output models is a linear relationship between outputs $k([x,d],[x',d']) = \sum_e P_{de}P_{d'e}k_e(x,x')$, where P is a (not necessarily square) matrix. This model is equivalent to multiplying the independent outputs with the matrix P which can be specified or learned through the marginal likelihood. This approach is developed in Chapter 4. It is also possible to use the same covariance function for each output $k_e(x,x') = k(x,x')$, which additionally simplifies computation.

Multi-output approaches have been used to model neural population [Byron et al., 2009] data, compiler performance [Bonilla et al., 2008], real-time sensor data [Osborne et al., 2008], a robot arm [Seeger et al., 2005] and time series forcasting [Boyle and Frean, 2005]. A review of multioutput approaches is given in Alvarez et al. [2012].

1.4 Learning covariance functions

A suitable covariance function may not be known in advance, and simple choices such as the examples in Figure 1.1 may be insufficient for real applications. There is a large body of literature concerning the data-driven construction of kernels to address this problem. This section reviews some kernel learning approaches.

The words 'inference' and 'learning' can take different meanings in the machine learning literature. Here, 'inference' is meant in the Bayesian sense. That is, combining a prior and a likelihood with Bayes' rule. 'Learning' refers to optimizing the parameters of a model. Under this distinction, the kernel learning approach does not add additional uncertainty to the model. Generally, the kernel learning approaches introduce parameters to the kernel, where the parameters do not have a prior distribution. Later, Section 1.6 considers the case where additional *probabilistic* components are combined with a Gaussian process, resulting in a non-Gaussian model.

Hyperparameter optimization A simple form of kernel learning is to learn hyperparameters of a given kernel. For example, stationary kernels, where $k(x,x') = \rho(||x-x'||)$ with $||x-x'||$ is a scaled squared distance $||x-x'||^2 = \sum_d (x_d - x'_d)^2 \ell_d^{-2}$, have parameters ℓ_d which determines the characteristic scale in each dimension. These parameters can be learned through marginal likelihood [Rasmussen and Williams, 2006] or cross validation [Stone, 1974]. The marginal likelihood approach has the

advantage of great simplicity (equation (1.8) is straightforward to implement), but the disadvantage is that it amounts to choosing the prior after seeing the data, so runs the risk of overfitting. If the number of hyperparameters is small compared to the number of data then this approach is likely to be a close approximation of a Bayesian approach [MacKay, 1994], however. Hyperparameter optimization via the marginal likelihood is extensively used and usually presented as a core part of the Gaussian process method Rasmussen and Williams [2006]. Cross-validation is less frequently used, perhaps due to its relative expense and difficulty of implementation. The leave-one-out estimator is most common option, though batch approaches have also been used Sundararajan and Keerthi [2000].

Compositional kernels Kernels can be combined in sums and products to obtain new richer kernels from simple base kernels. Additive kernels Duvenaud et al. [2011], Bach [2009] and more general compositions of sums and products Duvenaud et al. [2013], Lloyd et al. [2014] can express a rich covariance structure, using hierarchical combinations of base kernels such as those in Figure 1.1. The approach of Lloyd et al. [2014] builds combinations of sums and products of kernels using a discrete search. A great advantage of the discrete approach is that it is possible to express the result in terms of natural language, as each base kernel is interpretable by design, and sums and (to some extent) products of kernels are also interpretable. Later work has optimized over continuous hierarchical combinations of base kernels [Sun et al., 2018], which is likely to be faster and easier to implement, though less interpretable. This approach of [Sun et al., 2018] combines base kernels in a fixed hierarchy of multiple layers with learnable parameters at each layer.

Input warping Another approach is to compose a deterministic function with a simple kernel, using the fact that $k(f(x), f(x'))$ is always a valid kernel if k is a valid kernel [Rasmussen and Williams, 2006]. The function f could be arbitrarily complex and could be learned from data via the marginal likelihood. This approach was demonstrated to be effective in Wilson and Nickisch [2015], Wilson et al. [2016], Bradshaw et al. [2017], Al-Shedivat et al. [2017], Calandra et al. [2016], with f parametrized by a neural network. A drawback of the approach is that f is not regularized so is prone to overfitting. In particular, if there is great flexibility in f then mapping between $f(x_n)$ and y_n can be very simple and so the GP is not necessarily providing any modelling power. Bradshaw et al. [2017] has shown that the GP component can help prevent adversarial attacks, and Wilson et al. [2016] shows that performance is improved over the neural network alone.

Spectral methods Stationary kernels are functions of only distance between points, $k(x, x') = \rho(x - x')$, so can be analysed in the Fourier domain of ρ. Bochner's theorem states that there is a one-to-one correspondence between positive measures in the frequency domain and stationary kernel. This correspondence allows kernel to be readily constructed in the frequency domain. The frequency space can be estimated by interpolating the Fourier transform of the data [Sato et al., 2011]

or given a parametric form [Wilson and Adams, 2013, Samo and Roberts, 2015]. This approach is particularly well suited to data with periodic components, but in principle can be used for any stationary kernel. A non-stationary variant of the spectral approach was proposed by Remes et al. [2017].

Process convolution The process convolution approach constructs a new process by convolving a base Gaussian process (usually white noise) with some function g_x,

$$f(x) = \int g_x(u)\mathrm{d}H(u) \tag{1.30}$$

where H is the white noise process. Since convolution is a linear operation the resulting process is still a Gaussian process, with covariance

$$k(x, x') = \int g_x(u)g_{x'}(u)\mathrm{d}H(u) \tag{1.31}$$

Note that positive definiteness is guaranteed for arbitrary g_x and H. In fact, it is possible to define a kernel directly from (1.31) using some g_x and H, though the integral might not be tractable. A special choice that is tractable is $g_x(u) = \mathcal{N}(x|u, \frac{1}{2}\ell^2)$ and $H = 1$. For the moment we redefine the RBF kernel to have a factor of $\frac{1}{2}$ in the exponent for convenience with the Gaussian, so that the RBF kernel with lengthscale ℓ and variance σ^2 is given by $k_{\text{RBF}} = \sigma^2 \exp(-\frac{1}{2\ell^2}(x - x')^2) \propto \mathcal{N}(x|x', \ell^2)$. With this definition, the kernel of (1.31) recovers the RBF kernel with lengthscale ℓ. This can be shown using standard properties of Gaussians, as follows

$$\begin{aligned}
k(x, x') &= \int \mathcal{N}(x|u, \tfrac{1}{2}\ell^2)\mathcal{N}(x'|u, \tfrac{1}{2}\ell^2)\mathrm{d}u && (1.32)\\
&= \int \mathcal{N}(u|\tfrac{1}{2}(x + x'), \tfrac{1}{4}\ell^2)\mathcal{N}(x|x', \ell^2)\mathrm{d}u && (1.33)\\
&= \mathcal{N}(x|x', \ell^2) && (1.34)\\
&= \tfrac{1}{\sqrt{2\pi\ell^2}} \exp \tfrac{-1}{2\ell^2}(x - x')^2 && (1.35)\\
&= \sigma^2 \exp \tfrac{-1}{2\ell^2}(x - x')^2 \,. && (1.36)
\end{aligned}$$

The function g_x can be chosen to have inducing desired properties. For example, Álvarez and Lawrence [2011] derive the multiple output approach of 5.1.5 from the process convolution perspective, and Bruinsma and Turner [2018] incorporates causal information for time series data through the shape of g. Further work has placed a Gaussian process prior over g [Tobar et al., 2015], resulting in a non-Gaussian model. A non-linear convolution model was proposed in Álvarez et al. [2019], which again gives a non-Gaussian model.

Non-stationary kernels Any stationary kernel can be modified to a non-stationary variant [Paciorek, 2003]. The construction can first be demonstrated to derive a non-stationary RBF kernel by using $g_x(u) = \mathcal{N}(x|u, \frac{1}{2}\ell(x)^2)$ in the derivation of the

RBF kernel via convolution. With this choice, the derivation becomes

$$k(x,x') = \int \mathcal{N}(x|u, \tfrac{1}{2}\ell(x)^2)\mathcal{N}(x'|u, \tfrac{1}{2}\ell(x)^2)\mathrm{d}u \tag{1.37}$$

$$= \int \mathcal{N}(u|\lambda(\tfrac{2x}{\ell(x)^2} + \tfrac{2x'}{\ell(x')^2}), \lambda)\mathcal{N}(x|x', \tfrac{\ell(x)^2+\ell(x')^2}{2})\mathrm{d}u \tag{1.38}$$

$$= \mathcal{N}(x|x', \tfrac{\ell(x)^2+\ell(x')^2}{2}) \tag{1.39}$$

$$= \frac{1}{\sqrt{2\pi\frac{\ell(x)^2+\ell(x')^2}{2}}} \exp \frac{-(x-x')^2}{\ell(x)^2+\ell(x')^2} \tag{1.40}$$

$$\propto \frac{1}{\sqrt{\ell(x)^2+\ell(x')^2}} \exp \frac{-(x-x')^2}{\ell(x)^2+\ell(x')^2} \tag{1.41}$$

where $\lambda^{-1} = \frac{2}{\ell(x)^2} + \frac{2}{\ell(x')^2}$

This covariance function has non-constant diagonal as $k(x,x) = \frac{1}{2\ell(x)^2}$. To correct this, the covariance function can be rescaled as follows

$$k(x,x') = \sqrt{\frac{2\ell(x)\ell(x')}{\ell(x)^2+\ell(x')^2}} \exp \frac{-(x-x')^2}{\ell(x)^2+\ell(x')^2} \tag{1.42}$$

This is a non-stationary version of the RBF kernel, which was independently discovered by Gibbs [1998] and Paciorek [2003]. Paciorek [2003] further generalized this approach to modify any stationary kernel to a non-stationary variant, and proposed to use another GP for $\log \ell(x)$. The resulting model is non-Gaussian, and is the first example in the literature of a deep GP. To see the connection between input warping (i.e., where $k(x,x')$ is replaced with $k(f(x), f(x'))$ for some function f) and the non-stationary approach, the kernel input can be considered to be the concatenation of the input and the lengthscale, $[x, \ell]$, and the warping function f takes the form $f([x,\ell]) = [x, g(\ell)]$, where g is some positive function.

Infinite limits of deep networks Once of the catalysts for the renewed interest in Gaussian processes during the late 1990s was the observation that a (Bayesian) neural network with an single layer converged to a Gaussian process in the limit of infinite hidden units [Neal, 1996]. The result relies on the central limit theorem to show Gaussianty. In the case of a relu activation function [Cho and Saul, 2009] and error function [Williams, 1997] it is possible to compute the resulting covariance function in closed form. Cho and Saul [2009] went further to show - albeit by an erroneous argument [Matthews et al., 2018] - that a multilayer network with infinite width at each layer also results in a Gaussian process. The covariance function is tractable form for the relu activation. The multi-layer general case was revisited by Lee et al. [2018], Matthews et al. [2018], where it was shown that infinitely wide networks with appropriate priors do indeed converge to Gaussian processes. Lee et al. [2018] also proposed an approach to compute the covariance function approximately for cases other than the relu activation. Recently, Novak et al. [2019], Garriga-Alonso et al. [2019] generalized the results to convolutional networks.

It is interesting to consider the implications of these results for infinite width neural networks. As discussed by Matthews et al. [2018], the correspondence to Gaussian

processes could be considered a disadvantage, for reasons discussed in the next section. This suggests that neural networks should not be too wide. There is also a further issue that the covariance between the hidden units vanishes as the number of units goes to infinity. This is the source of the computational tractability, but is arguably a disadvantage as the weights cannot represent features as in the finite case [MacKay, 1998].

1.5 Limitations of Gaussian processes

"*It is important to keep in mind that Gaussian processes are not appropriate priors for all problems*" [Neal, 1998]. Even with a complicated kernel, the assumption of joint Gaussianity is limiting in many situations. While this section perhaps states the obvious, it is important to identify the limitations of Gaussian processes before seeking to overcome them.

Marginals The most obvious limitation is the assumption of Gaussian marginals. In particular, a Gaussian process can not model heavy-tailed, asymmetric or multi-modal marginal distributions. This point is the subject of Chapters 4 and 5, where a deep model will be used to generate non-Gaussian marginal densities. To create non-Gaussian marginals within the single layer framework the only available option is to appropriately modify the likelihood. This brings an immediate difficulty of choosing such a likelihood: a suitable form might not be known, and highly parameterized approaches incur the risk of overfitting if not carefully regularized. A standard approach for modelling heavy-tailed marginals is the student-t likelihood [Jylänki et al., 2011].

Covariance A second limitation of Gaussian process modelling is the requirement to define a prior entirely in terms of mean and covariance. Even very complicated covariance functions can not express prior relationships that are not captured by mean and covariance alone. Consider for example a very simple hierarchical model with two variables

$$p(a,b) = \mathcal{N}\left(\begin{pmatrix} a \\ b \end{pmatrix} \middle| \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right) \qquad \rho \sim \text{Bin}(\rho \in \{\tfrac{9}{10}, 0\}, p = \{\tfrac{1}{2}, \tfrac{1}{2}\})\,. \tag{1.43}$$

This model is certainly non-Gaussian and so cannot be represented with a GP, regardless of mean and covariance function. An example generalization of this situation to continuous data could be a GP with a lengthscale that could be one of two values. This is quite a natural model. A further generalization might be a continuous prior on the lengthscale, such as a Gamma distribution. At the risk of belabouring the point, this example demonstrates that it is not possible to use a GP to represent all prior information: even quite simple a natural seeming model is not representable by a GP.

The further generalization of the example above is a non-stationary process with a characteristic lengthscale that is unknown and varies across the space. An extreme example of this is a process with discontinuities at unknown locations. This cannot be represented by a GP [Neal, 1996]. Note that a discontinuity at a known location can readily modelled by a GP (for example, $k(x, x') = xx'$ if $\text{sign}(x) = \text{sign}(x')$ and 0 otherwise, has a discontinuity at zero). A discontinuity at an *unknown* location cannot be represented in this way, however.

There is a body of literature devoted to the infinite data limit of GPs. An important result is that GPs with certain kernels (e.g. the RBF kernel) can fit any function in the limit of infinite data [see, for example van der Vaart et al., 2008]. This 'universal representation' is often cited as an important property that justifies the use of a GP. These results are quite orthogonal to the issue of what prior information can be expressed by a GP.

Predictive covariance decoupling The predictive covariance of the exact posterior process is independent of the observed outputs and is only a function of the inputs (see equation 1.12). It is unsatisfactory that the predictive uncertainty is decoupled from the observations. For example, if data is observed on a grid then the predictive distribution also follows a grid structure, irrespective of output values.

To contrive a geological example: when measuring rock density it might be known that a certain light rock always is found in large deposits spanning a large area, whereas the heavy rock is found in tightly localised deposits. If a measurement is made and it is found to be light, then the predictive uncertainty at nearby locations should smaller be than if it had been found to be heavy. This effect cannot be captured by a Gaussian process, even with a sophisticated non-stationary covariance function.

As a further example, consider the experiment from Garnelo et al. [2018b], where regression is performed on the pixels of the image of a face (the inputs are the pixel coordinates and the outputs the pixel grey-scale intensity). The face contains many structures (eyes, nose, teeth, etc.) for which we have strong prior information, but the precise location of the structures is not known in an (unaligned) image. If a given pixel is selected and it is found to be a certain colour then the predictive uncertainty should certainly change. For example, suppose it is found to be a white colour, so the pixel could be part of teeth or eyes, but not part of the nose. In this case uncertainty of nearby pixels should change to reflect the information (for example that the nose is not there), but this cannot happen with a Gaussian process.

Linearity of predictive mean The predictive mean of a Gaussian process with Gaussian observation model is linear in the observations (see equation (1.11)). A GP can be interpreted as a kernel smoother [Rasmussen and Williams, 2006], which is perhaps not a very interesting model. There are many situations where this linearity might be inappropriate, for example if the output gives structural information, as in the face example above. It should be stressed that a GP model can never capture these sort of effects, regardless of covariance function.

1.6 Approaches to overcome the limitations of GPs

There are many approaches to overcome the limitations above by departing from the Gaussian process model. Many of them are Bayesian versions of the approach to constructing more complex kernels, and many combine more than one GP in the same model. This section describes some of the approaches that depart from Gaussianity while still keeping a Gaussian process as part of the model. In some cases the models are equivalent or closely related to the deep GP introduced in the next chapter.

Hyperpriors Placing a prior over a hyperparameter of a kernel results in a non-Gaussian model which may better reflect the underlying phenomenon. Approximate inference is required, and typically inference is designed to harness some of the analytic properties of a GP. There is an extensive literature on methods for approximate inference in Gaussian process models with hyperpriors. Examples include Gibbs sampling [Titsias et al., 2008], slice sampling [Murray and Adams, 2010], Hamiltonian Monte-Carlo [Matthews et al., 2016a] and Sequential Monte Carlo [Svensson et al., 2015]. These models can give rise to non-Gaussian effects such as multimodal posterior predictive distributions. If the hyperparameters are local ones, which is generally the case, then these approaches cannot capture local effects such as discontinuities at known locations.

Input warping The Bayesian approach to input warping has been used in Bayesian optimization to achieve a non-stationary model [Snoek et al., 2014]. Bayesian optimization is often used to optimize hyperparameter choices for expensive machine learning approach such as neural networks, where it may takes weeks to evaluate a single hyperparameter configuration. Hyperparameter sensitivity is generally considered to be non-stationary [Snoek et al., 2014], and since data is scarce it is important to use a Bayesian approach. The approach of Snoek et al. [2014] uses a beta function CDF as a warping over each dimension.

Using a Gaussian process for the warping is known as a deep GP, which is the subject of the next and subsequent chapters.

Output warping The approach of Snelson et al. [2004] warps a GP with a parametric function that is fit via marginal likelihood. This approach results in a non-Gaussian model. The warping in Snelson et al. [2004] is fixed across the space, which is potentially a restrictive assumption. To achieve a warping that can change across the space, Lázaro-Gredilla [2012] used another GP for the warping function. This is another example of a deep GP and is a special case of the models introduced in the next chapter.

Non-stationary covariances The non-stationary approach of Gibbs [1998], Higdon et al. [1999] requires a parametric form for the lengthscale at each point. Paciorek

[2003] used another GP for this (log) lengthscale function, resulting in a particular instance of a deep GP. This model was developed further by Heinonen et al. [2016] to include GPs for the kernel log variance and likelihood log variance. This non-stationary deep GP model can in fact be considered a special case of the deep GP introduced in the next chapter, and was recently proposed in the multi-layer case by Dunlop et al. [2017], Monterrubio-Gómez et al. [2018], Salimbeni and Deisenroth [2017a], Remes et al. [2017].

Other processes The Student-t process can be derived from a scale mixture of Gaussian process distributions [O'Hagan, 1991, O'Hagan et al., 1998]. The Student-t process shares some analytic properties with Gaussian processes, though is more robust to outliers and has been used for Bayesian optimization [Tracey and Wolpert, 2018, Shah et al., 2014]. The neural processes [Garnelo et al., 2018a,b] is a neural network model for functions which is learned through maximum likelihood rather than Bayesian inference. The dependency of test points on training data can be non-linear though the use of non-linear neural networks, and the output distribution can be made complex by forming a density by passing a Gaussian variable through a non-linear function. The neural process model requires training samples from the desired function class. This is in contrast to the Bayesian approach where the function class is specified through a prior distribution. The Gaussian process regression network [Wilson et al., 2011] is an extension of the multi-output approach described in 5.1.5, but where the mixing matrix has components given by a Gaussian process, rather than being fixed. In a related approach Adams and Stegle [2008] form heteroscedastic model by multiplying a GP with the exponential of another GP.

This chapter has introduced the technical machinery that forms the foundation of later chapters, and has discussed the limitations of single layer GP models and approaches to overcome them. The next chapter introduces the deep Gaussian process, which can be thought of as a Bayesian non-parametric approach to kernel learning.

Chapter 2

Doubly Stochastic Variational Inference for Deep Gaussian Processes

Gaussian processes (GPs) are a good choice for function approximation as they are flexible, robust to overfitting, and provide well-calibrated predictive uncertainty. As the previous chapter has argued, however, standard GPs are limited in many ways. One considerable limitation is that simple kernels cannot effectively model complex data and expressive kernels either require domain knowledge or need to be inferred. Inference over kernels is expensive or incurs the risk of over-fitting if done approximately. Deep Gaussian processes (DGPs) are multi-layer generalizations of GPs that promise to overcome some of the limitations of the single layer model. Inference in these models has proved challenging, however. Existing approaches to inference in DGP models assume approximate posteriors that force independence between the layers. The chapter presents a doubly stochastic variational inference algorithm that does not force independence between layers. With this method of inference it is demonstrated that a DGP model can be used effectively on data ranging in size from hundreds to a billion points. This chapter also considers the DGP prior and suggests that the use of linear mean functions for inner layers to overcome issues of degeneracy.

In common with many state-of-the-art GP approximation schemes, the starting point of this chapter is the sparse inducing point variational framework introduced in Chapter 1. Differently from previous work, the inference presented in this chapter does not force *selectional* independence between the layers. Selectional dependence between layers introduces an additional source of analytic intractability. This difficulty forces the use of a sampling approach, introducing the first source of stochasticity. This is computationally straightforward due to an important property of the sparse

variational posterior marginals: the univariate marginals conditioned on the layer below depend only on the corresponding inputs. It follows that samples from the marginals at the top layer can be obtained without computing the full covariance within the layers. For large data applications the data is further subsampled in minibatches. This second source of stochasticity allows the approach to scale to arbitrarily large data.

Experiments are presented that demonstrate that the approach works well in practice. Results are provided on benchmark regression and classification data problems, including the first application of a DGP to a dataset with a billion points. The experiments confirm that DGP models are never worse than single-layer GPs, and in many cases significantly better. Crucially, it is shown that additional layers do not incur overfitting, even with small data.

2.1 Related Work

Early examples of models where the outputs of a GP used as the inputs to another GP can be found in Lawrence and Moore [2007] and Paciorek and Schervish [2004]. In Lawrence and Moore [2007] the RBF kernel is used for both layers and the input to the second GP is the output of the first, and a MAP was used for inference. In Paciorek and Schervish [2004] the second layer is a GP with non-stationary Matern kernel (similar to (1.42)), where the lengthscale is given by the exponential of the output of the first GP. A sampling approach was used for inference.

Using the model of Lawrence and Moore [2007], the seminal work of Titsias and Lawrence [2010b] demonstrated how sparse variational inference could be used to propagate Gaussian inputs through a GP with a Gaussian likelihood. This approach was extended in Damianou et al. [2011] to perform approximate inference in the model of Lawrence and Moore [2007], and shortly afterwards in a similar model **?**, which also included a linear mean function. The key idea of both these approaches is the factorization of the variational posterior *between* layers. A more general model (flexible in depth and dimensions of hidden layers) was introduced by Damianou and Lawrence [2013a] and coined the term 'DGP'. This approach used a posterior that also factorized between layers and required a linearly increasing number of variational parameters in the number of data. For high-dimensional observations, it is possible to amortize the cost of this optimization with an auxiliary model. This approach is pursued in Dai et al. [2016], and with a recurrent architecture in Mattos et al. [2016]. Another approach to inference in the exact model was presented in Hensman and Lawrence [2014], where a sparse approximation was used within layers for the GP outputs, similar to Damianou and Lawrence [2013b], but with a projected distribution over the inputs to the next layer. The particular form of the variational distribution was chosen to admit a tractable bound, but imposes a constraint on the flexibility.

An alternative approach is to modify the DGP prior directly and perform inference in a parametric model. This is achieved in Bui et al. [2016] with an inducing point

approximation within each layer, and in Cutajar et al. [2017] with an approximation to the spectral density of the kernel. Both approaches then apply additional approximations to achieve tractable inference. In Bui et al. [2016], an approximation to expectation propagation is used, with additional Gaussian approximations to the log partition function to propagate uncertainly through the non-linear GP mapping. In Cutajar et al. [2017] a fully factorized variational approximation is used for the spectral components. Both these approaches require specific kernels: in Bui et al. [2016] the kernel must have analytic expectations under a Gaussian, and in Cutajar et al. [2017] the kernel must have an analytic spectral density. Vafa [2016] also uses the same initial approximation as Bui et al. [2016] but applies MAP inference for the inducing points, such that the uncertainty propagated through the layers only represents the quality of the approximation. In the limit of infinitely many inducing points this approach recovers a deterministic radial basis function network. A particle method is used in Wang et al. [2016], again employing an online version of the sparse approximation used by Bui et al. [2016] within each layer. Similarly to the approach presented here, in Wang et al. [2016] samples are taken through the conditional model, but they then use a point estimate for the latent variables. It is not clear how this approach propagates uncertainty through the layers, since the GPs at each layer have point-estimate inputs and outputs.

A pathology with the DGP with zero mean function for the inner layers was identified in **?**. In **?** a suggestion was made to concatenate the original inputs at each layer. This approach is followed in Dai et al. [2016] and Cutajar et al. [2017]. Notably in Cutajar et al. [2017] a 30 layer DGP model is successfully trained. A linear mean function was original used by **?**, though in the special case of a two layer DGP with a 1D hidden layer.

A special case of the DGP has a white noise process (where all GP variables are independent a priori) for the inner layer(s). This approach is known as the Gaussian process latent variable model [Lawrence, 2004] and the Gaussian process with a latent covariate model [Wang and Neal, 2012], and is explored in detail in Chapters 4 and 5. The original presentation of the DGP in Damianou and Lawrence [2013a] principally models where some or all of the layers have white noise kernels rather than the fully noise-free models considered in this chapter. Discussion of the role of noise in the DGP model is deferred to Chapter 5.

As briefly mentioned in the first paragraph of this section, an alternative construction of a multi-layer GP is through non-stationary kernels. Rather than transform the input directly, this approach uses a kernel with a per-point lengthscale. The non-stationary kernel was originally proposed in Higdon et al. [1999] and then generalized in Paciorek [2003]. In Paciorek [2003] a two layer model is proposed with a sampling-based scheme for inference. The non-stationary approach was used in Heinonen et al. [2016] with Hamiltonian Monte Carlo. The multi-layer extension where each but the first layer is a non-stationary GP with lengthscale given by (a positive transform of) the output of the layer below, was explored in Dunlop et al. [2017], Salimbeni and Deisenroth [2017a], Monterrubio-Gómez et al. [2018]. In Dunlop et al. [2017] it is shown that the infinite depth limit is non-trivial, which is in contrast to the compositional approach of Damianou and Lawrence [2013a] which is developed here.

The non-stationary construction is amenable to the methods presented in this chapter [Salimbeni and Deisenroth, 2017a], but is not explored further in this thesis.

2.2 Deep Gaussian Processes

There are many ways to construct a deep Gaussian process (DGP). Four such constructions are given in Dunlop et al. [2017] but for the moment we consider the approach of [Damianou and Lawrence, 2013b]. In this construction independent GPs $f^1, \dots, f^L$ are combined through function composition. The composite function g is given by $g : x \mapsto f^L(f^{L-1}(\dots f^1(x))))$. Given a likelihood $p(y|g(x))$, the DGP model of Damianou and Lawrence [2013a] has the joint density,

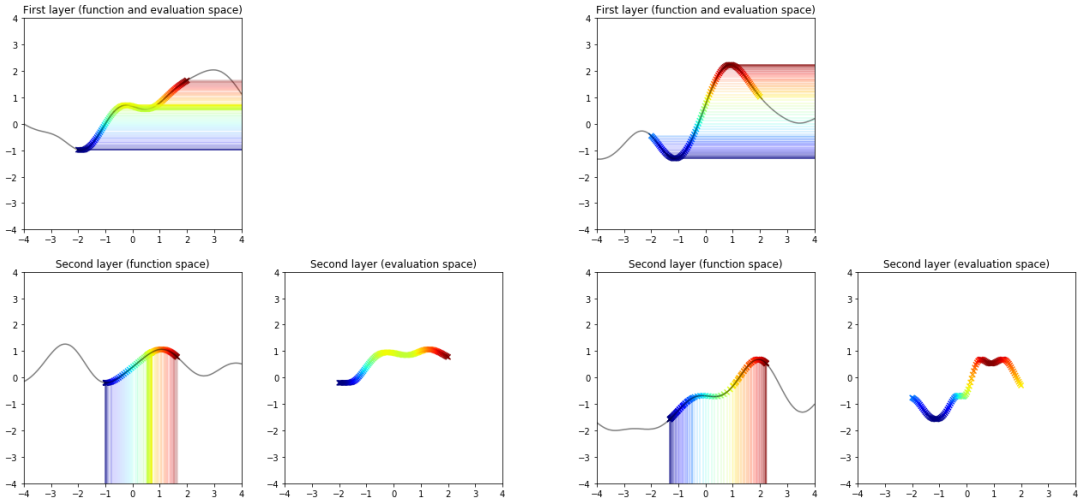
$$p(y, \{f^l\}_{l=1}^L) = \underbrace{\prod_{i=1}^N p(y_i|f^L(f^{L-1}(\dots f^1(x_i))))}_{\text{likelihood}} \underbrace{\prod_{l=1}^L p(f^l)}_{\text{prior}}, \tag{2.1}$$

where each layer is a GP $p(f^\ell) = \mathcal{GP}(m^\ell, k^\ell)$.

Salimbeni and Deisenroth [2017b] describe this model as a 'recursive prior' where the outputs of each layer are the inputs to the next. This description confuses *statistical* and *selectional* relationships. These terms were introduced in Chapter 1: a *statistical* relationship is determined through the joint density of two given (fixed) variables, whereas a *selectional* relationship is where the association is introduced by the selection of a variable. In the case of the DGP in (2.1) the priors for each layer are independent by construction, yet $f^1(x)$ and $f^2(f^1(x))$ are *selectionally* dependent.

The *statistical* and *selectional* distinction is made clear when visualizing the DGP. Figure 2.1 shows prior samples from two perspectives. In the first row a sample from f^1 is shown (smooth curve) together with $f_n^1 = f(x_n)$ (crosses), where x_n are 50 points equally spaced in the interval $[-2, 2]$. The colours indicate the output values (i.e. the value of f_n^1). In the second row there are two plots. On the left is the *statistical* perspective where f^2 is plotted (smooth curve) together with $f_n^2 = f^2(f_n^1)$ at inputs f_n^1. On the right plot is the *selectional* perspective where f_n^2 is plotted against x_n. The graphical model for the DGP is shown in Figure 2.2 together with the likelihood evaluation diagram. The evaluation diagram makes use of the f_n^2 and f_n^1 notation, but it should be emphasized that these should not be thought of as random variables.

Figure 2.1 reveals a 'pathology' with the DGP that was first identified in Duvenaud et al. [2014]. In a single dimension it is clear that a zero mean GP prior is highly non-injective. A demonstration of a 'fold' in the first layer mapping can be seen in Figure 2.1(a). When this happens a symmetry occurs at the next layer in the evaluation space. This is potentially undesirable. In higher dimensions the problem is reduced but still present. It can be shown that in the limit of infinite layers this model collapses to a trivial limit [Dunlop et al., 2017, Duvenaud et al., 2014]. One remedy to this problem is to define the second layer inputs in an augmented space



(a) Example with few folds

(b) Example with two folds

Figure 2.1: Two visulations of the two layer DGP prior, indicating the *statistical* perspective (left) and *selectional* perspective (right). The smooth curves are samples from the functions, whereas the crosses are points being propagated through the layers in the likelihoood.

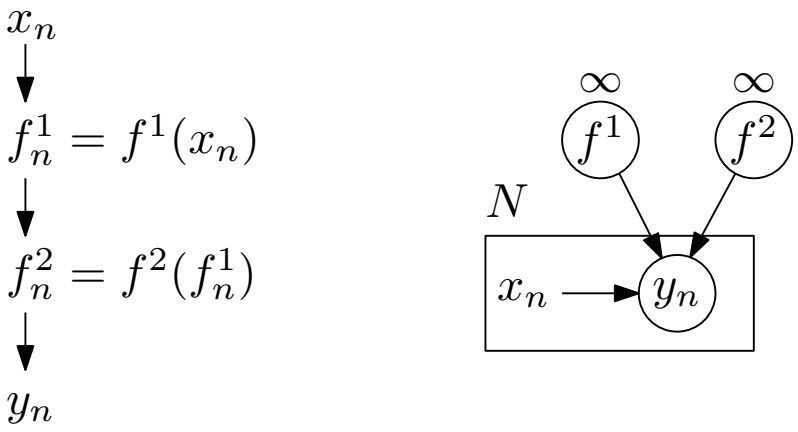
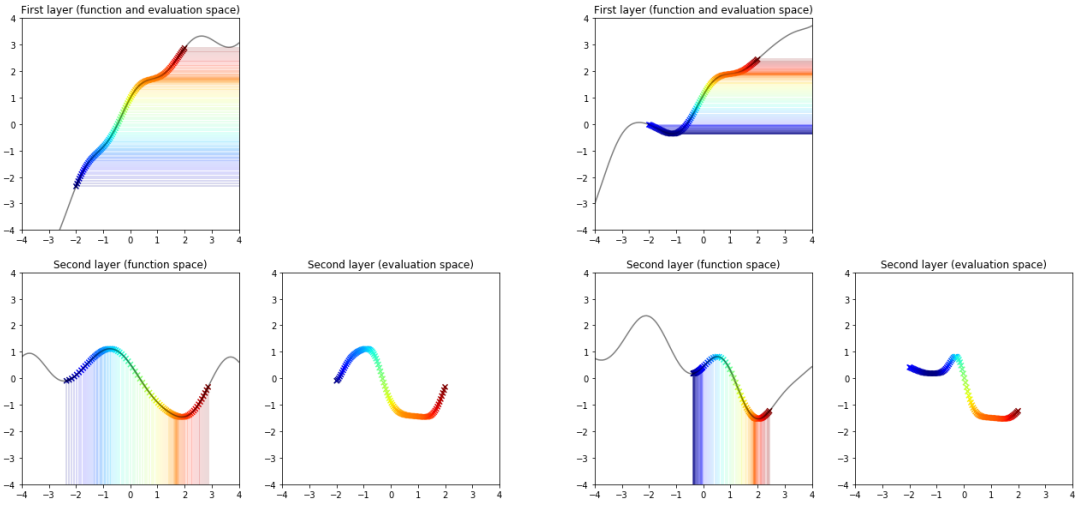


Figure 2.2: A two layer DGP model depicted graphically. Right: the probabilistic graphical model. Left: the likelihood evaluation diagram.



(a) Example with no folds

(b) Example with one fold

Figure 2.3: As Figure 2.1 but with the identity mean function for the first layer. The chance of folds is reduced.

formed by concatenating the original inputs with the layer outputs. This approach was considered by Cutajar et al. [2017] and Dai et al. [2016].

As an alternative, Salimbeni and Deisenroth [2017b] proposed to use a identity mean function at each inner layer. Figure 2.3 shows samples from a two layer model with the identity mean function for the first layer. There is a small fold in 2.3(b), but the chance of a large fold is reduced. When the inner layer dimensionality is different Salimbeni and Deisenroth [2017b] proposes to use a fixed linear mean function using the PCA components of the data. This gives the best low-rank approximation to the identity mapping. The use of mean function in this was inspired by the 'skip layer' approach of the ResNet architecture of He et al. [2016].

Models so far have considered scalar inputs and outputs for each layer. In practice, x_n will be vector valued and the inner layers can also be vector valued. A GP is only defined for scalar outputs, but it is possible to extend a GP to higher dimensional outputs by considering the 'flattened' representation, as discussed in Section 1.3. In this chapter, all multioutput layers will use independent outputs with shared covariance functions. As the extension from scalar to independent outputs is straightforward it will not be notated explicitly.

2.3 Inference

This section presents the novel variational posterior originally proposed by Salimbeni and Deisenroth [2017b], and demonstrates a method to obtain unbiased samples from the resulting lower bound. The inference is based on the approach by Matthews [2017], introduced in Chapter 1. The key idea is to consider inference over the entire process, not just the N variables associated with data. This approach avoids

the difficulty of representing the finite variables in the likelihood. The approach is demonstrated by the statistical/selectional distinction of Figures 2.1 and 2.3: inference is concerned with the functions in the left column, not the crosses that appear in the likelihood.

The two layer case is presented here for the purposes of illustrating the key idea of inference while keeping notation as light as possible. The extension to multiple layers is straightforward. In this case the joint density is

$$p(y, f^1, f^2) = \underbrace{\prod_{i=1}^N p\left(y_i | f^2(f^1(x_i))\right)}_{\text{likelihood}} \underbrace{p(f^1)p(f^2)}_{\text{prior}} . \tag{2.2}$$

The variational posterior chosen is mean-field between the layers:

$$q(f^1, f^2) = q(f^1)q(f^2) . \tag{2.3}$$

The variables f^ℓ can be partitioned into a finite set $\tilde{\mathbf{f}}^\ell$, where $[\tilde{\mathbf{f}}^\ell]_m = f^\ell(\tilde{x}_m^\ell)$ for $m = 1, \ldots, M^\ell$ and all the remaining variables f_*^ℓ, for $\ell = 1, 2$.

Following the idea from the single layer case [**?**Matthews, 2017], a variational posterior can be formed for each

$$q(f^\ell) = p(f_*^\ell | \tilde{\mathbf{f}}^\ell) q(\tilde{\mathbf{f}}^\ell) . \tag{2.4}$$

That is, the variational posterior in (2.3) is assumed to be a product of two sparse posteriors of the form (1.16). The choice of the finite part of these posteriors is taken to be Gaussian

$$q(\tilde{\mathbf{f}}^\ell) = \mathcal{N}(\mathbf{m}^\ell, \mathbf{S}^\ell) . \tag{2.5}$$

With these choices, the variational posteriors are themselves GPs. The mean and covariance functions are exactly as in the single layer case. That is, $q(f^\ell) = \mathcal{GP}(\mu_{\mathbf{m}^\ell}, \Sigma_{\mathbf{S}^\ell})$, with

$$\mu_{\mathbf{m}^\ell} = m^\ell(x) + \mathbf{k}^\ell(x)^\top \mathbf{K}^{\ell^{-1}} \mathbf{m}^\ell \tag{2.6}$$

$$\Sigma_{\mathbf{S}^\ell}(x, x') = k(x, x') + \mathbf{k}^\ell(x)^\top \mathbf{K}^{\ell^{-1}} (\mathbf{S}^\ell - \mathbf{K}^\ell) \mathbf{K}^{\ell^{-1}} \mathbf{k}^\ell(x') . \tag{2.7}$$

Substituting the model (2.2) and variational posterior (2.4) into the general form for the ELBO (1.14) and applying the same partitioning for the priors for f and g yields

$$\mathcal{L}_q = \prod_{n=1}^N p\left(y_i | f^2(f^1(x_n))\right) - \text{KL}(q(f^1)||p(f^1)) - \text{KL}(q(f^2)||p(f^2)) . \tag{2.8}$$

Both KL terms reduce to computations over the inducing point variables, for identical reasoning as the single layer case described in 1.2. Making this simplification, and

using properties of the logarithm, the bound simplifies to

$$\mathcal{L}_q = \sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(f^1)q(f^2)} \log p\left(y_i|f^2(f^1(x_i))\right)}_{=L_i} - \mathrm{KL}(q(\tilde{\mathbf{f}}^1)||p(\tilde{\mathbf{f}}^1)) - \mathrm{KL}(q(\tilde{\mathbf{f}}^2)||p(\tilde{\mathbf{f}}^2)) \tag{2.9}$$

The difficulty presented by the deep model is how to evaluate L_i. In the single layer case there is only an expectation under the univariate marginals of a GP, which can be performed using 1D quadrature [Matthews et al., 2016b], sampling [Gal et al., 2015, Krauth et al., 2016b] and analytically [Hensman et al., 2013]. Fortunately, the requirement to only compute the marginal propagates through the expectation in the multilayer case. To see this, first consider the $q(f^1)$ expectation. Only the variable $f^1(x_i)$ appears in the expectation, so the expectation over $q(f^1)$ can be replaced by an expectation over $q(f^1(x_i))$, which is a 1D Gaussian. This expectation can be reparameterized in terms of $\epsilon^1 \sim \mathcal{N}(0,1)$ as follows

$$\begin{aligned}
L_i &= E_{q(f^2)q(f^1)} \log p\left(y_i|f^2(f^1(x_i))\right) && (2.10)\\
&= E_{q(f^2)p(f^1(x_i))} \log p\left(y_i|f^2(f^1(x_i))\right) && (2.11)\\
&= E_{q(f^2)p(\epsilon^1)} \log p\left(y_i|f^2(\mu_{\mathbf{m}^1}(x_i) + \epsilon^1\sqrt{k_{\mathbf{S}^1}(x_i,x_i)}\right) && (2.12)\\
&= E_{q(f^2)p(\epsilon^1)} \log p\left(y_i|f^2(z_i(\epsilon^1))\right) && (2.13)
\end{aligned}$$

where $z_i(\epsilon^1) = \mu_{\mathbf{m}^1}(x_i) + \epsilon^1\sqrt{k_{\mathbf{S}^1}(x_i,x_i)}$. Similarly, the expectation over $q(f^2)$ can similarly be replaced by an expectation over $q(f^2(z_i(\epsilon^1)))$.

$$\begin{aligned}
L_i &= E_{q(f^2)p(\epsilon^1)} \log p\left(y_i|f^2(z_i(\epsilon^1))\right) && (2.14)\\
&= E_{q(f^2(z_i(\epsilon^1)))p(\epsilon^1)} \log p\left(y_i|f^2(z_i(\epsilon^1))\right) && (2.15)\\
&= E_{p(\epsilon^2)p(\epsilon^1)} \log p\left(y_i|f^2(z_i(\epsilon^1))\right) && (2.16)\\
&= E_{p(\epsilon^2)p(\epsilon^1)} \log p\left(y_i|\mu_{\mathbf{m}^2}(z_i(\epsilon^1)) + \epsilon^2\sqrt{k_{\mathbf{S}^2}(z_i(\epsilon^1),z_i(\epsilon^1))}\right) && (2.17)
\end{aligned}$$

Writing the L_i term in this way shows two crucial properties. First, that the expectation is over two independent scalar variables. Second, that both variables can be reparameterized in terms of standard Gaussians (knows as the 'reparameterization trick' [Kingma and Welling, 2013]). While the expectation is non-linear in both ϵ^1 and ϵ^2 it is possible to obtain an unbiased estimate by taking a Monte Carlo estimate. That is, L_i can be approximated as

$$L_i \approx \frac{1}{S}\sum_{s=1}^{S} \log p\left(y_i|\mu_{\mathbf{m}^2}(z_i(\epsilon_s^1)) + \epsilon_s^2\sqrt{k_{\mathbf{S}^2}(z_i(\epsilon_s^1),z_i(\epsilon_s^1))}\right), \quad \epsilon_s^1, \epsilon_s^2 \sim \mathcal{N}(0,1) \tag{2.18}$$

The bound is a sum over the L_i, which can be estimated using data subsampling,

$$\mathcal{L}_q \approx \frac{N}{|B|}\sum_{i\in B} L_i - \mathrm{KL}(q(\tilde{\mathbf{f}}^1)||p(\tilde{\mathbf{f}}^1)) - \mathrm{KL}(q(\tilde{\mathbf{f}}^2)||p(\tilde{\mathbf{f}}^2)) \tag{2.19}$$

where B is a batch of data. There are therefore two sources of stochasticity: the Monte Carlo estimate of each L_i, and the data sampling. Both stochastic approximations are unbiased and have variance that can be driven to zero with additional computation.

In the case of multiple outputs the estimation of L_i is performed with each dimension independently. The true posterior will have couplings between the outputs but the variational posterior makes the assumption of independence over the outputs so the estimation is straightforward.

For predictions we are primarily interested in the distribution of $f^2_* = f^2(f^1(x_*))$. The true posterior is given by

$$f^2_* = \mathbb{E}_{p(f^1,f^2|y)} f^2(f^1(x_*))\,, \tag{2.20}$$

which can be approximated by

$$f^2_* \approx \mathbb{E}_{q(f^1)q(f^2)} f^2(f^1(x_*))\,, \tag{2.21}$$

where the expectation is approximated using the parametrization and sampling exactly as in (2.17).

The bound is differentiable with respect to both the variational parameters ($\tilde{\mathbf{m}}^\ell$, $\tilde{\mathbf{S}}^\ell$ and $\{\tilde{x}^\ell\}_m$) and also the model parameters (kernel hyperparameters and any likelihood parameters). Optimizing the bound with respect to model parameters incurs a bias due to the non-uniform slack in the lower bound to the true marginal likelihood. Nonetheless, this approach appears to work well and in all experiments the bound is simultaneously optimized with respect to all model and variational parameters (the mean functions remain fixed in the experiments in this chapter).

2.4 Results

We evaluate our inference method on a number of benchmark regression and classification datasets. We stress that we are interested in models that can operate in both the small and large data regimes, with little or no hand tuning. All our experiments were run with exactly the same hyperparameters and initializations (all lengthscales and variances were initialized to 1, the learning rate of the Adam optimizer was 0.01 and the batch size $10,000$). We use $\min(30, D^0)$ for all the inner layers of our DGP models, where D^0 is the input dimension, and the RBF kernel for all layers.

We compare with another inference method for the DGP model [**?**], and to Bayesian neural networks with probabilistic back-propagation for inference [Hernández-Lobato and Adams, 2015]. As a baseline, we compare to a single layer GP model with the same number of inducing points (100 in for all models, initialized with K−means).

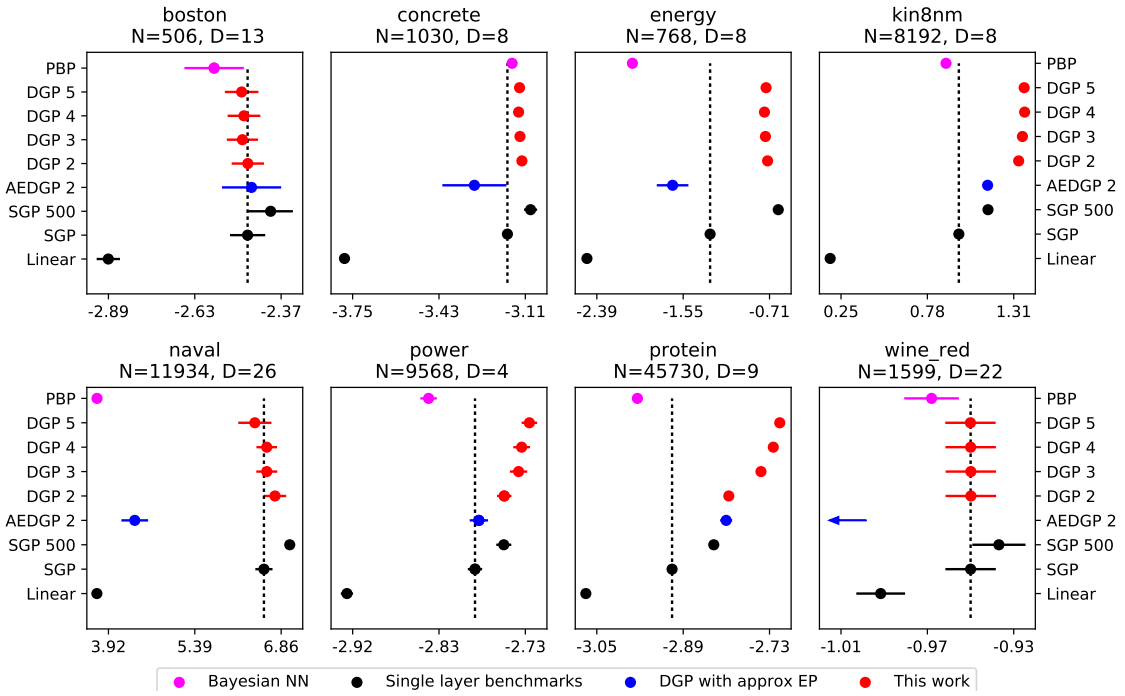


Figure 2.4: Regression test log-likelihood results on benchmark datasets. Higher (to the right) is better. The sparse GP with the same number of inducing points is highlighted as a baseline.

To assess the quality of sparse approximation we compare also to a sparse GP with five times the inducing points. We also ran experiments on GP models with neural network kernels Wilson et al. [2016], Calandra et al. [2016], but found that these models overfit significantly on the smaller datasets (they achieved sub-linear results on all datasets with fewer than 10K points), so they are not a relevant comparison.

Regression Benchmarks We compare our approach to other state-of-the-art methods on 8 standard small to medium-sized UCI benchmark datasets. Following common practice [e.g. Hernández-Lobato and Adams, 2015] we use 20-fold cross validation with a 10% randomly selected held out test set and scale the inputs and outputs to zero mean and unit standard deviation within the training set (we restore the output scaling for evaluation). While we could use any kernel, we choose the RBF kernel with a lengthscale for each dimension for direct comparison with Bui et al. [2016].

The test log-likelihood results are shown in Fig. 2.4. We compare our models of 2, 3, 4 and 5 layers (DGP 2–5), each with 100 inducing points, with (stochastically optimized) sparse GPs [Hensman et al., 2013] with 100 and 500 inducing points points (SGP, SGP 500). We emphasize that the single layer model with 500 inducing point took significantly longer to train than the DGP models (see Table 2.4 for a timing comparison).

We compare also to a two-layer Bayesian neural network with ReLu activations, 50 hidden units (100 for protein and year), with inference by probabilistic back-

propagation [Hernández-Lobato and Adams, 2015] (PBP). The results are taken from Hernández-Lobato and Adams [2015] and were found to be the most effective of several other methods for inferring Bayesian neural networks. We compare also with a DGP model with approximate expectation propagation (EP) for inference [Bui et al., 2016]. Using the authors' code[1] we ran a DGP model with 1 hidden layer using approximate expectation propagation [Bui et al., 2016] (AEPDGP 2). We used the input dimension for the hidden layer for a fair comparison with our models[2]. We found the time requirements to train a 3-layer model with this inference prohibitive.

On five of the eight datasets, the deepest DGP model is the best. On 'wine', 'naval' and 'boston' our DGP recovers the single-layer sparse GP, which is not surprising: 'boston' is very small, 'wine' is near-linear (note the proximity of the linear model and the scale) and 'naval' is characterized by extremely high test likelihoods (the RMSE on this dataset is less than 0.001 for all SGP and DGP models), i.e. it is a very 'easy' dataset for a GP. The Bayesian network is not better than the sparse GP for any dataset and significantly worse for six. The Approximate EP inference for the DGP models is also not competitive with the sparse GP for many of the datasets, but this may be because the initializations were designed for lower dimensional hidden layers than we used.

Our results on these small and medium sized datasets confirm that overfitting is not observed with the DGP model, and that the DGP is never worse and often better than the single layer GP. We note in particular that on the 'power', 'protein' and 'kin8nm' datasets all the DGP models outperform the SGP with five times the number of inducing points.

Rectangles Benchmark We use the Rectangle-Images dataset[3], which is specifically designed to distinguish deep and shallow architectures. The dataset consists of 12,000 training and 50,000 testing examples of size 28×28, where each image consists of a (non-square) rectangular image against a different background image. The task is to determine which of the height and width is greatest. We run 2, 3 and 4 layer DGP models, and observe increasing performance with each layer. Table 2.1 contains the results. Note that the 500 inducing point single-layer GP is significantly less effective than any of the deep models. Our 4-layer model achieves 77.9% classification accuracy, exceeding the best result of 77.5% reported in Larochelle et al. [2007] with a three-layer deep belief network. We also exceed the best result of 76.4% reported in Krauth et al. [2016a] using a sparse GP with an Arcsine kernel, a leave-one-out objective, and 1000 inducing points.

Large-Scale Regression To demonstrate our method on a large scale regression problem we use the UCI 'year' dataset and the 'airline' dataset, which has been commonly used by the large-scale GP community. For the 'airline' dataset we take

[1] `https://github.com/thangbui/deepGP_approxEP`

[2] We note however that in Bui et al. [2016] the inner layers were 2D, so the results we obtained are not directly comparable to those reported in Bui et al. [2016]

[3] `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/RectanglesData`

Table 2.1: Results on Rectangles-Images dataset ($N = 12000$, $D = 784$)

	Single layer GP		Ours			Larochelle [2007]		Krauth [2016]
	SGP	SGP 500	DGP 2	DGP 3	DGP 4	DBN-3	SVM	SGP 1000
Accuracy (%)	76.1	76.4	77.3	77.8	**77.9**	77.5	76.96	76.4
Likelihood	−0.493	−0.485	0.475	**−0.460**	**−0.460**	-	-	−0.478

the first 700K points for training and next 100K for testing. We use a random 10% split for the 'year' dataset. Results are shown in Table 2.2, with the log-likelihood reported in 2.3. In both datasets we see that the DGP models perform better with increased depth, significantly improving in both log likelihood and RMSE over the single-layer model, even with 500 inducing points.

Table 2.2: Regression test RMSE results for large datasets

	N	D	SGP	SGP 500	DGP 2	DGP 3	DGP 4	DGP 5
year	463810	90	10.67	9.89	9.58	8.98	8.93	**8.87**
airline	700K	8	25.6	25.1	24.6	24.3	24.2	**24.1**
taxi	1B	9	337.5	330.7	281.4	270.4	268.0	**266.4**

Table 2.3: Regression test log likelihood results for large datasets

	N	D	SGP	SGP 500	DGP 2	DGP 3	DGP 4	DGP 5
year	463810	90	−3.74	−3.65	−3.63	−3.57	−3.56	**−3.32**
airline	700K	8	−4.66	−4.63	−4.61	−4.59	−4.59	**−4.58**
taxi	1B	9	−7.24	−7.22	−7.06	−7.02	−7.01	**−7.00**

MNIST Multiclass Classification We apply the DGP with 2 and 3 layers to the MNIST multiclass classification problem. We use the robust-max multiclass likelihood [Hernández-Lobato et al., 2011] and use full unprocessed data with the standard training/test split of 60K/10K. The single-layer GP with 100 inducing points achieves a test accuracy of 97.48% and this is increased to 98.06% and 98.11% with two and three layer DGPs, respectively. The 500 inducing point single layer model achieved 97.9% in our implementation, though a slightly higher result for this model has previously been reported of 98.1% [Hensman et al., 2013] and 98.4% [Krauth et al., 2016a] for the same model with 1000 inducing points. We attribute this difference to different hyperparameter initialization and training schedules, and stress that we use exactly the same initialization and learning schedule for all our models. The only other DGP result in the literature on this dataset is 94.24% [Wang et al., 2016] for a two layer model with a two dimensional latent space. State-of-the-art for permutation invariant MNIST classification is 99.1% for a support vector machine with learned features from a deep network [Tang, 2013].

Table 2.4: Typical computation time in seconds for a single gradient step. All models have 100 inducing points, except SGP 500, which has 500

	CPU	GPU
SGP	0.14	0.018
SGP 500	1.71	0.11
DGP 2	0.36	0.030
DGP 3	0.49	0.045
DGP 4	0.65	0.056
DGP 5	0.87	0.069

Large-Scale Classification We use the HIGGS ($N = 11 \times 10^6$, $D = 28$) and SUSY ($N = 5.5 \times 10^6$, $D = 18$) datasets for large-scale binary classification. These datasets have been constructed from Monte Carlo physics simulations to detect the presence of the Higgs boson and super-symmetry [Baldi et al., 2014]. We take a 10% random sample for testing and use the rest for training. We use the AUC metric for comparison with Baldi et al. [2014]. Our DGP models are the highest performing on the SUSY dataset (AUC of 0.877 for all the DGP models) compared to shallow neural networks (NN, 0.875), deep neural networks (DNN, 0.876) and boosted decision trees (BDT, 0.863). On the HIGGS dataset we see a steady improvement in additional layers (0.830, 0.837, 0.841 and 0.846 for DGP 2–4 respectively). On this dataset the DGP models exceed the performance of BDT (0.810) and NN (0.816) and both single layer GP models SGP (0.785) and SGP 500 (0.794). The best performing model on this dataset is a 5 layer DNN (0.885).

Massive-Scale Regression To demonstrate the efficacy of our model on massive data we use the New York city yellow taxi trip dataset of 1.21 billion journeys [4]. Following Peng et al. [2017] we use 9 features: time of day; day of the week; day of the month; month; pick-up latitude and longitude; drop-off latitude and longitude; travel distance. The target is to predict the journey time. We randomly select 1B (10^9) examples for training and use 1M examples for testing, and we scale both inputs and outputs to zero mean and unit standard deviation in the training data. We discard journeys that are less than 10 s or greater than 5 h, or start/end outside the New York region, which we estimate to have squared distance less than 5^o from the center of New York. The test RMSE results are the bottom row of Table 2.2 and test log likelihoods are in Table 2.3. We note the significant jump in performance from the single layer models to the DGP. As with all the large-scale experiments, we see a consistent improvement extra layers, but on this dataset the improvement is particularly striking (DGP 5 achieves a 21% reduction in RMSE compared to SGP)

[4] `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`

2.5 Conclusion

Our experiments show that on a wide range of tasks the DGP model with our doubly stochastic inference is both effective and scalable. Crucially, we observe that on the small datasets the DGP does not overfit, while on the large datasets additional layers generally increase performance and never deteriorate it. In particular, we note that the largest gain with increasing layers is achieved on the largest dataset (the taxi dataset, with 1B points). We note also that on all the large scale experiments the SGP 500 model is outperformed by *all* the DGP models. Therefore, for the same computational budget increasing the number of layers can be significantly more effective than increasing the accuracy of approximate inference in the single-layer model. Other than the additional computation time, which is fairly modest (see Table 2.4), we do not see downsides to using a DGP over a single-layer GP, but substantial advantages. While we have considered simple kernels and black-box applications, any domain-specific kernel could be used in any layer. This is in contrast to other methods [Damianou and Lawrence, 2013b, Bui et al., 2016, Cutajar et al., 2017] that require specific kernels and intricate implementations. The number of optimized hyperparameters in our model is small: we have a vector of lengthscales for each layer and a scalar variance, and a likelihood variance hyperparameter, so $L(D+1)+1$ parameters in total, where D is the input dimension. Our implementation is simple (< 200 lines), publicly available [5], and is integrated with GPflow [Matthews et al., 2017], an open-source GP framework built on top of Tensorflow [Abadi et al., 2015].

[5] `https://github.com/ICL-SML/Doubly-Stochastic-DGP`

Chapter 3

Natural Gradients in Practice: Non-Conjugate Variational Inference in Gaussian Process Models

This chapter considers the problem of optimizing variational parameters. The previous chapters have left the problem of finding variational parameters as a black-box optimization problem in both the single layer and deep GP models. The difficulty of this problem is dependent on choice of optimizer, initialization, and parameterization of the model. The optimization has some special structure that can be exploited by use of the natural gradient, however. The natural gradient method has been used effectively in conjugate Gaussian process models, but the non-conjugate case has been largely unexplored. This chapter examines in detail how natural gradients can be used in non-conjugate stochastic settings, together with hyperparameter learning. It is shown that the natural gradient can significantly improve performance in terms of wall-clock time. For ill-conditioned posteriors the benefit of the natural gradient method is especially pronounced, and we demonstrate a practical setting where ordinary gradients are unusable. We show how natural gradients can be computed efficiently and automatically in any parameterization, using automatic differentiation.

3.1 Introduction

Minimizing the Kullback-Leibler (KL) divergence between an unknown and a tractable parametric distribution is the central task of variational inference. In the non-conjugate case, the prevalent approach is to optimize the objective using (stochastic)

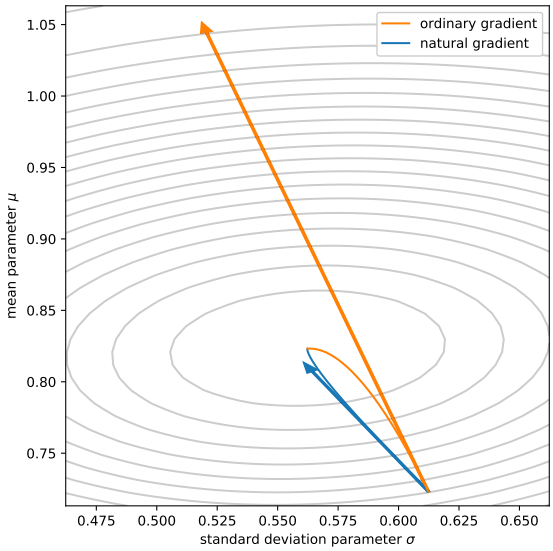


Figure 3.1: The natural gradient (blue arrow) is correctly scaled and points in a better direction than the ordinary gradient (orange arrow). The contours show the lower bound to a GP with a Bernoulli likelihood and variational posterior with a single Gaussian inducing point. The path followed by taking very small steps in the ordinary gradient (orange curve) ascends the contours. The path taking small steps in the natural gradient (blue curve) is independent of parameterization, and does not follow the contours.

gradient descent or variants. Gradient descent based methods require careful tuning to work effectively and are prone to poor convergence when the Hessian at the solution is ill-conditioned [Boyd and Vandenberghe, 2004]. Ill-conditioning is a problem especially encountered in kernel methods [Ma and Belkin, 2017]. A further problem is that the step size is not dimensionless but its units are in the square of the parameters. The appropriate step size is tightly coupled with the parameterization, and no best step size can exist for all problems.

The ordinary gradient turns out to be an unnatural direction to follow for variational inference since we are optimizing a *distribution*, rather than a set of parameters directly. One way to define the gradient is the direction that achieves maximum change subject to a perturbation within a small euclidean ball. To see why the euclidean distance is an unnatural metric for probability distributions, consider the two Gaussians $\mathcal{N}(0, 0.1)$ and $\mathcal{N}(0, 0.2)$, compared to $\mathcal{N}(0, 1000.1)$ and $\mathcal{N}(0, 1000.2)$. The former pair are different and the latter similar, yet in euclidean distance they are equally far apart in the mean and variance. Using the precision in place of the variance gives the opposite result, yet the distributions are unchanged. There is a fundamental mismatch between the ordinary gradient and the objective function: the gradient is dependent on parameterization whereas the objective function is not.

Fortunately there is a way to solve the disparity: the natural gradient. The natural gradient can be defined as the direction that achieves maximum change in KL divergence. It is well known that paths following the natural gradient are invariant to reparameterization [see e.g., Martens, 2014], and that the natural gradient direction is the ordinary gradient rescaled by the inverse Fisher information matrix [Amari,

1998]. Fig. 3.1 shows a two parameter example comparing the natural gradient to the ordinary gradient. In this case we see that the natural gradient points in a better direction than the ordinary gradient, and also has an appropriate scale.

To investigate whether the advantages suggested by Fig. 3.1 hold in practice, we consider several aspects in turn. We begin by comparing the different gradients across several different parameterizations (Section 3.4.1). To achieve this we first demonstrate how natural gradients can be calculated efficiently and without any cumbersome derivations. Through empirical investigation we show that the natural gradient is indeed a more effective direction to follow in all parameterizations, and also that there is an appropriate step size to use after an initial phase. Using these insights we propose a gradient descent algorithm for the common situation where the size of the dataset forces us to subsample the data, leading to stochastic gradients (Section 3.4.2). We then extend the approach to hyperparameter optimization using a double loop algorithm that outperforms the state-of-the-art Adam optimizer in wall-clock time (Section 3.4.3). Finally, we demonstrate a situation where natural gradients are essential for successful optimization, due to ill-conditioning (Section 3.4.4).

The contributions of this chapter are as follows:

- We compare natural gradients to other state-of-the-art techniques in non-conjugate problems, explicitly comparing the influence of different parameterizations and step size on performance.
- We show how natural gradients in the exponential family can be computed efficiently and automatically in any parameterization, using automatic differentiation.
- We show that natural gradients can be used in conjunction with hyperparameter learning in the stochastic setting.
- We highlight a situation where the current approaches fail due to ill-conditioning, and show that natural gradients can solve this problem.
- In the DGP case we show that natural gradients can be used effectively for the final layer and also for all layers.

3.2 Background

In this section we introduce the relevant background on the exponential family, variational inference, and optimisation approaches. We then define natural gradients and show how they take a simple form for the exponential family.

3.2.1 Preliminaries

We consider the problem of performing inference in a model of the form

$$p(\mathbf{y}, \mathbf{u}) = \left[\prod_{n=1}^{N} p(y_n|\mathbf{u})\right] p(\mathbf{u})\,, \tag{3.1}$$

where y_n are observed and $\mathbf{u}$ unobserved. Both the prior and likelihood may additionally depend on hyperparameters, but we have omitted these from the notation to reduce clutter. We assume that exact inference in (3.1) is intractable and make use of an approximate posterior $q(\mathbf{u};\boldsymbol{\theta})$ in the exponential family. The exponential family is defined as

$$\log q(\mathbf{u};\boldsymbol{\theta}) = \log h(\mathbf{u}) + \boldsymbol{\theta}^\top \mathbf{t}(\mathbf{u}) - A(\boldsymbol{\theta})\,, \tag{3.2}$$

where $\mathbf{t}(\mathbf{u})$ is the sufficient statistics vector, $A(\boldsymbol{\theta})$ is the log normalizing constant and $h(\,\cdot\,)$ is a base measure. The parameterization used in (3.2) is known as the *natural* parameterization and $\boldsymbol{\theta}$ are the natural parameters. We can instead use an alternative smooth invertible parameterization $\boldsymbol{\xi} \equiv \boldsymbol{\xi}(\boldsymbol{\theta})$. We denote transformation between parameterizations through overloading notation, e.g., the inverse mapping back to the natural parameterization is $\boldsymbol{\theta}(\boldsymbol{\xi})$, and we also abbreviate $q(\mathbf{u};\boldsymbol{\theta}(\boldsymbol{\xi}))$ as $q(\mathbf{u};\boldsymbol{\xi})$. A parameterization of particular interest, known as the *expectation* parameterization, is defined as $\boldsymbol{\eta} \equiv \mathbb{E}_{q(\mathbf{u};\boldsymbol{\theta})}\mathbf{t}(\mathbf{u})$. An important property of the exponential family is that the gradient of the log normalizer is equal to the expectation parameter: $\nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) = \boldsymbol{\eta}^\top$. This can be readily identified from differentiating (3.2) with respect to $\boldsymbol{\theta}$ and taking expectations.[1] Differentiating (3.2) again, it follows that the Hessian of the log density is a Jacobian:

$$\nabla^2_{\boldsymbol{\theta}} \log q(\mathbf{u};\boldsymbol{\theta}) = -\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}}\,. \tag{3.3}$$

Variational inference proceeds by minimizing the KL divergence from $q(\mathbf{u};\boldsymbol{\xi})$ to the intractable posterior $p(\mathbf{u}|\mathbf{y})$, or equivalently maximizing the evidence lower bound (ELBO):

$$\mathcal{L}(\boldsymbol{\xi}) = \mathbb{E}_{q(\mathbf{u};\boldsymbol{\xi})} \sum_{n=1}^{N} \log p(y_n|\mathbf{u}) - \mathrm{KL}[q(\mathbf{u};\boldsymbol{\xi})||p(\mathbf{u})]\,. \tag{3.4}$$

Our fundamental problem is to minimize $-\mathcal{L}(\boldsymbol{\xi})$. All the approaches we consider find a sequence of parameters $\{\boldsymbol{\xi}_t\}_{t=0}^T$ using the iterative update

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \gamma_t \mathbf{P}_t^{-1}\mathbf{g}_t, \quad \mathbf{g}_t = \nabla_{\boldsymbol{\xi}^\top}\mathcal{L}\Big|_{\boldsymbol{\xi}=\boldsymbol{\xi}_t}\,, \tag{3.5}$$

where γ_t denotes the *step size* and $\mathbf{P}_t^{-1}\mathbf{g}_t$ the *direction*.

[1] Differentiating (3.2): $\nabla_\theta \log q(\mathbf{u};\boldsymbol{\theta}) = \mathbf{t}(\mathbf{u})^\top - \nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta})$. Taking expectations: $\mathbb{E}_{q(\mathbf{u};\boldsymbol{\xi})}\nabla_{\boldsymbol{\theta}} \log q(\mathbf{u};\boldsymbol{\theta}) = \boldsymbol{\eta}^\top - \nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta})$. The score has expectation zero, so the result follows.

3.2.2 Optimization approaches

Gradient descent (GD). The simplest approach, known as gradient descent, is to set $\mathbf{P}$ to the identity matrix. The step size can be fixed, decayed, or found by a line search on each iteration.

Adam A more sophisticated approach is to use a diagonal matrix for $\mathbf{P}$, with diagonal elements given by $(\sqrt{v_i}+\epsilon)^{-1}m_i$, where m_i and v_i are the bias corrected exponential moving averages of $[\mathbf{g}_t]_i$ and $([\mathbf{g}_t]_i)^2$. This approach is called Adam [Kingma and Ba, 2015].

LBFGS One way of interpreting the update (3.5) is to identify the term $\mathbf{P}_t^{-1}\mathbf{g}_t$ as a minimizer of the local quadratic approximation $\mathcal{L}(\boldsymbol{\xi}_t+\boldsymbol{\delta}) \approx L(\boldsymbol{\xi}_t)+\mathbf{g}^\top\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^\top\mathbf{P}\boldsymbol{\delta}$. Under this interpretation, a natural choice for $\mathbf{P}$ is the Hessian so that the quadratic approximation coincides with the second order Taylor expansion. This is known as Newton's method. Due to the large computational expense of calculating and inverting the Hessian we do not consider it further. Instead, we use a low rank approximation to the Hessian computed from finite differences. Specifically we will compare to a common variant of this algorithm known as LBFGS [Byrd et al., 1995]. This algorithm cannot be used in the stochastic setting as finite difference calculations are not robust to noise.

Natural gradient descent (NGD) Another way of interpreting the update (3.5) is to use the fact that the direction of steepest descent with respect to a norm $||\boldsymbol{\delta}||_{\mathbf{A}} = \boldsymbol{\delta}^T\mathbf{A}\boldsymbol{\delta}$ is given by $\mathbf{A}^{-1}\nabla_{\boldsymbol{\xi}^\top}\mathcal{L}$.[2] Identifying $\mathbf{P}$ with $\mathbf{A}$, the update (3.5) corresponds to the steepest descent with respect to the norm induced by the matrix $\mathbf{P}$. Gradient descent (where $\mathbf{P}$ is the identity and the induced metric is Euclidean) can therefore be seen as moving in the direction that maximizes the change in objective with respect to the euclidean norm of the parameters. The Euclidean norm is an unnatural way to compare two parameter vectors if the parameters correspond to *distributions*, however. If instead we consider the KL divergence between two distributions and take the small perturbation limit, we obtain $\mathrm{KL}[q(\mathbf{u};\boldsymbol{\xi}), q(\mathbf{u};\boldsymbol{\xi}+\boldsymbol{\delta})] = \frac{1}{2}\boldsymbol{\delta}^\top\left[\mathbb{E}_{q(\mathbf{u};\boldsymbol{\xi})}\nabla^2_{\boldsymbol{\xi}}\log q(\mathbf{u};\boldsymbol{\xi})\right]\boldsymbol{\delta}+\mathcal{O}(||\boldsymbol{\delta}||^3)$. Therefore, in a sufficiently small neighbourhood the KL divergence induces a quadratic norm with curvature given by the expected Hessian of the log density. This matrix is known as the Fisher information $\mathbf{F}_{\boldsymbol{\xi}}$,

$$\mathbf{F}_{\boldsymbol{\xi}} = -\mathbb{E}_{q(\mathbf{u};\boldsymbol{\xi})}\nabla^2_{\boldsymbol{\xi}}\log q(\mathbf{u};\boldsymbol{\xi})\,. \tag{3.6}$$

The direction of steepest descent with respect to this norm is called the natural gradient $\tilde{\nabla}_{\boldsymbol{\xi}}\mathcal{L}$, given by the gradient scaled by the inverse Fisher information: $\tilde{\nabla}_{\boldsymbol{\xi}}\mathcal{L} = (\nabla_{\boldsymbol{\xi}}\mathcal{L})\,\mathbf{F}_{\boldsymbol{\xi}}^{-1}$ [Amari, 1998].

[2] This can be seen by minimizing $\frac{1}{\epsilon}\mathcal{L}(\boldsymbol{\xi}+\boldsymbol{\delta})$ subject to the constraint that $||\boldsymbol{\delta}||_{\mathbf{A}} = \epsilon$ and letting $\epsilon \to 0$.

For the exponential family the Fisher information takes a particularly simple form in the natural parameters. Using (3.3) we have that $\mathbf{F}_{\boldsymbol{\theta}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}}$. Using the chain rule, we see that the natural gradient in the natural parameters is given by $\tilde{\nabla}_{\boldsymbol{\theta}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}}$. This expression was used by Hensman et al. [2013] to compute natural gradients in the conjugate case.

To find the natural gradients in some other paramterization we can use the chain rule to obtain

$$\mathbf{F}_{\boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}}\right)^\top \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}} \,. \tag{3.7}$$

This expression was used directly in [Malagò and Pistone, 2015] and [Sun et al., 2009] in a certain parameterization of the Gaussian. The calculation is extremely cumbersome and requires a careful recursive implementation. In the next section we show how to compute the natural gradient efficiently and automatically.

3.2.3 Efficient computation

Since all the parameterizations are invertible (and the inverse of the Jacobian is the Jacobian of the inverse), we have

$$\tilde{\nabla}_{\boldsymbol{\xi}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} \left(\left(\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}}\right)^\top \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}} \right)^{-1} \tag{3.8}$$

$$= \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}} \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\theta}}\right)^\top . \tag{3.9}$$

Applying the chain rule and transposing, we obtain

$$\tilde{\nabla}_{\boldsymbol{\xi}^\top} \mathcal{L} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\theta}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}^\top} \,. \tag{3.10}$$

We recognise (3.10) as a Jacobian-vector product, which is exactly what is computed in *forward-mode* differentiation. Forward-mode automatic differentiation libraries are perhaps less common than reverse-mode, but fortunately there is an elegant way to achieve forward-mode automatic differentiation using reverse-mode differentiation twice [Townsend et al., 2017], explained in 3.2.4. Importantly, this indirect computation only costs negligibly more than the forward pass $\boldsymbol{\xi}(\boldsymbol{\theta})$. The extra computation comes from the parameter conversion between $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$, which is $\mathcal{O}(M^3)$ for the (full rank) Gaussian for the six parameterizations we consider in the next section, where M is the dimension of $\mathbf{u}$. Note that direct inversion of the Fisher information for the Gaussian would be cubic in the number of parameters, i.e., $\mathcal{O}((M + M^2)^3)$. In practice, we find this increases the computation relative to the ordinary gradient by a factor of about 1.5. We emphasize that this approach requires no more code than the parameter transformation, so new parameterizations can be easily investigated.

3.2.4 The forward-mode trick

Forward-mode derivatives can be obtained using a reverse-mode library. The trick is due to Townsend et al. [2017] and this explanation closely follows `https://j-towns.github.io/2017/06/12/A-new-trick.html`. Reverse-mode differentiation is the successive application of the vector-Jacobian product (vjp) operation. The vjp operation left multiplies a vector $\mathbf{u}$ with the Jacobian of $\mathbf{f}$ with respect to its input $\mathbf{x}$:

$$\text{vjp}(\mathbf{f}, \mathbf{x}, \mathbf{u}) = \mathbf{u}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \sum_i u_i \frac{\partial f_i}{\partial \mathbf{x}} \,.$$

The vjp operation can be used to implement the gradient of a function $L(\mathbf{f}(\mathbf{g}(\mathbf{x})))$ by using the chain rule $\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{f}} \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ and successively applying the vjp operation from left to right, i.e.,

$$\begin{aligned} \mathbf{u} &= \text{vjp}(L, \mathbf{f}, 1) \,, \\ \mathbf{u} &\leftarrow \text{vjp}(\mathbf{f}, \mathbf{g}, \mathbf{u}) \,, \\ \mathbf{u} &\leftarrow \text{vjp}(\mathbf{g}, \mathbf{x}, \mathbf{u}) \,. \end{aligned}$$

After these operations $\mathbf{u} = \frac{\partial L}{\partial \mathbf{x}}$. Automatic reverse-mode differentiation libraries implement vjp for all basic operations they support. Compositions of basic operations can be computed as above. Note that the values of $\mathbf{f}$ and $\mathbf{g}$ need to be computed first, which requires a forward pass through the function.

Forward-mode differentiation makes use of a Jacobian-vector product operation (jvp), defined as

$$\text{jvp}(\mathbf{f}, \mathbf{x}, \mathbf{u}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{u} = \sum_i \frac{\partial \mathbf{f}}{\partial x_i} u_i \,.$$

Using the jvp operation, the chain rule can be implemented by successive application of jvp, working from right to left, i.e.,

$$\begin{aligned} \mathbf{u} &= \text{jvp}(\mathbf{g}, \mathbf{x}, \mathbf{1}) \,, \\ \mathbf{u} &\leftarrow \text{jvp}(\mathbf{f}, \mathbf{g}, \mathbf{u}) \,, \\ \mathbf{u} &\leftarrow \text{jvp}(L, \mathbf{f}, \mathbf{u}) \,, \end{aligned}$$

where $\mathbf{1}$ is a vector of ones with the same shape as $\mathbf{x}$.

To implement natural gradients in any parameterization we require the jvp operation, but common libraries such as Tensorflow implement only vjp (i.e. reverse mode). The trick to achieve jvp from vjp is to introduce a dummy variable $\mathbf{v}$ and define $\mathbf{g}(\mathbf{v}) = \text{vjp}(\mathbf{f}, \mathbf{x}, \mathbf{v})$. We then use vjp again to find the gradient of $\mathbf{g}$ with respect to $\mathbf{v}$, passing in the vector $\mathbf{u}$ to be pushed forward: $\text{vjp}(\mathbf{g}, \mathbf{v}, \mathbf{u})$. Since $\mathbf{g}$ is linear in $\mathbf{v}$,

we have

$$\text{vjp}(\mathbf{g}, \mathbf{v}, \mathbf{u}) = \mathbf{u}^\top \frac{\partial}{\partial \mathbf{v}} \left(\mathbf{v}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) = \mathbf{u}^\top \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^\top .$$

This is exactly the transpose of $\text{jvp}(\mathbf{f}, \mathbf{x}, \mathbf{u})$. Therefore, any reverse-mode differentiation library can be used to compute forward-mode derivatives.

3.3 Specific application: sparse Gaussian processes

What we have described so far applies to any model and any exponential family variational posterior. We now present a specific example: a sparse Gaussian process (GP) model with a Gaussian variational posterior, introduced in Chapter 1. The model is reviewed here.

The model takes the form of (3.1). Each y_i is associated with a D-dimensional input $x_i \in \mathbb{R}^D$. We place a GP prior on the unobserved variables $f(x_i)$,

$$f(\,\cdot\,) \sim \mathcal{GP}(\mu, k)\,, \tag{3.11}$$

where μ and k are mean and covariance functions. That is, any collection of function values $f(x_1), \ldots, f(x_N)$ are jointly Gaussian with mean $\mu(x_i)$ and covariance $k(x_i, x_j)$, for $i, j = 1, \ldots, N$. Inference in this model scales cubically in N, and is intractable when the likelihood is not Gaussian, so we proceed with variational inference. We choose a Gaussian process for the posterior with the special property that it matches the prior conditioned on a number of inducing points $\mathbf{u} = [f(\mathbf{z}_i)]_{i=1}^M$ (NB the notation $\tilde{\mathbf{f}}$ was used for this quantity in Chapter 1, but $\mathbf{u}$ is used here lighten notation). We use a directly parameterized Gaussian for $q(\mathbf{u})$. The posterior leads to the bound,

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{u})} \sum_{i=1}^{N} \log \tilde{p}(y_i|\mathbf{u}) - \text{KL}[q(\mathbf{u})||p(\mathbf{u})]\,, \tag{3.12}$$

where $\log \tilde{p}(y_i|\mathbf{u}) = \mathbb{E}_{q(f_i|\mathbf{u})} \log p(y_i|f_i)$. Since both expectations are over Gaussians they combine to a single expectation with mean and variance available in closed form. The univariate expectation of the likelihood can be found with Gauss-Hermite quadrature (which is likely to be reasonably accurate due to the smoothness, see Figure 1.4), or exactly in some cases. The bound (3.12) can be evaluated stochastically by evaluating a random subset of terms in the sum and scaling the KL term appropriately.

Parameterizations of the Gaussian. We now present the different parameterizations we will use of the Gaussian variational distribution $q(\mathbf{u})$. The Gaussian is a member of the exponential family with the sufficient statistic vector given by $\mathbf{t}(\mathbf{u}) = [\mathbf{u}, \text{vec}(\mathbf{u}\mathbf{u}^\top)]$, so that $\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{u}) = \mathbf{u}^\top \boldsymbol{\theta}_1 + \mathbf{u}^\top \boldsymbol{\Theta}_2 \mathbf{u}$, where $\boldsymbol{\theta}_1$ is the first D elements of $\boldsymbol{\theta}$, and $\boldsymbol{\Theta}_2$ are remaining elements reshaped to a square matrix. We refer to this as the *unpacked form*. A common parameterization of the Gaussian is in terms of the mean ($\mathbf{m}$) and variance ($\mathbf{S}$). The unpacked natural parameters are given by

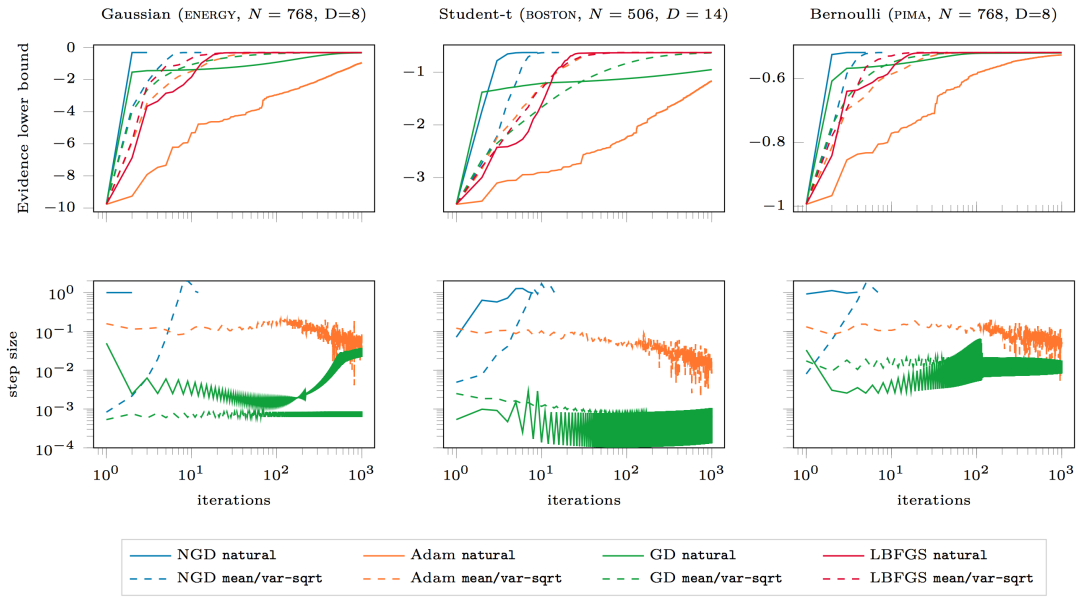


Figure 3.2: The natural gradient is a superior *direction* in both parameterizations, and the best step size increases during optimization to $\gamma \approx 1$. Upper row: optimization methods, all with a line search for the step size. Lower row: the step sizes used at each iteration.

$\mathbf{S}^{-1}\mathbf{m}, -\frac{1}{2}\mathbf{S}^{-1}$ and the expectation parameters by $\mathbf{m}, \mathbf{S}+\mathbf{m}\mathbf{m}^\top$. Converting between these parameterizations is straightforward and has complexity $\mathcal{O}(M^3)$, where M is the dimensionality of the Gaussian.

We consider six parameterizations of the Gaussian. Perhaps the most commonly used in variational inference [e.g., Dai et al., 2016, Challis and Barber, 2011] is the mean and square root of the covariance: $\mathbf{m}, \mathbf{L}$, with $\mathbf{L}\mathbf{L}^T = \mathbf{S}$. We refer to this as the `mean/var-sqrt` parameterization. This was the parameterization used in the previous chapter. Another way to constrain the covariance to be positive definite is to use the matrix log of the covariance [e.g., Glasmachers et al., 2010] $\mathbf{m}, \mathbf{L}$, with $\exp(\mathbf{L}) = \mathbf{S}$, where exp here is the matrix exponential (the `mean/var-log` parameterization). We use additionally the unconstrained mean and variance parameters $\mathbf{m}, \mathbf{S}$ (the `mean/var` parameterization) and the unconstrained natural parameters $\boldsymbol{\theta}_1, \boldsymbol{\Theta}_2$ (the `natural` parameterization). For completeness we also constrain the natural parameters via the square root and log transformations. Since $\boldsymbol{\Theta}_2$ is negative definite we use $\boldsymbol{\theta}_1, \mathbf{L}$ with $\mathbf{L}\mathbf{L}^T = -\boldsymbol{\Theta}_2$ (the `natural-sqrt` parameterization), and, finally, $\boldsymbol{\theta}_1, \mathbf{L}$ with $\exp(\mathbf{L}) = -\boldsymbol{\Theta}_2$ (the `natural-log` parameterization).

3.4 Natural gradients in practice

In this section we investigate NGD for large step sizes. We aim to provide evidence to answer the following:

1. Is the natural gradient a good *direction*, irrespective of step size?

2. Can we easily choose an effective step size?

3. Are natural gradients useful when combined with hyperparameter optimization?

We use a running example of three common datasets with different likelihoods: energy efficiency (ENERGY, $N = 784, D = 8$) with a Gaussian likelihood, boston housing (BOSTON, $N = 506, D = 14$) with a student-t likelihood, and pima Indians diabetes (PIMA, $N = 784, D = 8$) with a Bernoulli likelihood. We use 100 inducing points initialized with k-means and the Matern ($\nu = 5/2$) kernel.

3.4.1 Deterministic case

To investigate the quality of direction, we apply NGD, GD and Adam each with a line search to find the γ that achieves maximum value of the objective at each step. We run an exhaustive search for γ using the Brent [Brent, 1971] method until convergence. We compare also to LBFGS which includes a line search. Fig. 3.2 shows a representative split with the `mean/var-sqrt` and `natural` parameterizations. For the case of the Gaussian likelihood (Fig 3.2, left column) we observe the optimal solution is found in a single step of $\gamma = 1$, as shown in Hensman et al. [2013]. For the other parameterizations the initial natural gradient step size is a small value that is parameterization and likelihood dependent, but then increases to $\gamma = 1$. Once the step size has increased to near $\gamma = 1$ we observe extremely rapid convergence.

The natural gradient direction achieves faster convergence for all likelihoods and parameterizations.For all different parameterizations and likelihoods we see that the best step size for the natural gradient *increases* to $\gamma = 1$. This is in contrast to the ordinary gradient where the step size differs between likelihoods and generally needs to *decrease* as optimization progresses. For Adam the direction is elementwise rescaled and a value close to 0.1 seems appropriate for the constrained parameterizations, but for the unconstrained parameterizations Adam cannot make good progress.

In summary we have provided evidence that the natural gradient is indeed a better direction, and increasing the step size to $\gamma = 1$ is appropriate for fast convergence. We see also that the best combination of optimization method and parameterization is `natural` for NGD and `mean/var-sqrt` for GD and Adam. We will use these combinations for all subsequent experiments.

3.4.2 Stochastic natural gradients

We next consider the stochastic case where a line search is not possible. We introduce stochasticity by subsampling the data into minibatches of size 256. To find a reasonable γ for the Adam and GD methods we performed a search over $\{10^{-k}\}_{k=0}^{6}$.We used the largest rate that remained stable.

We now consider a strategy for γ. Our line search experiments suggest that γ should be gradually increased to some fixed value $\gamma \approx 1$. We therefore propose a simple schedule for NGD: (i) log-linearly increase γ from γ_{initial} to γ_{final} over K iterations;

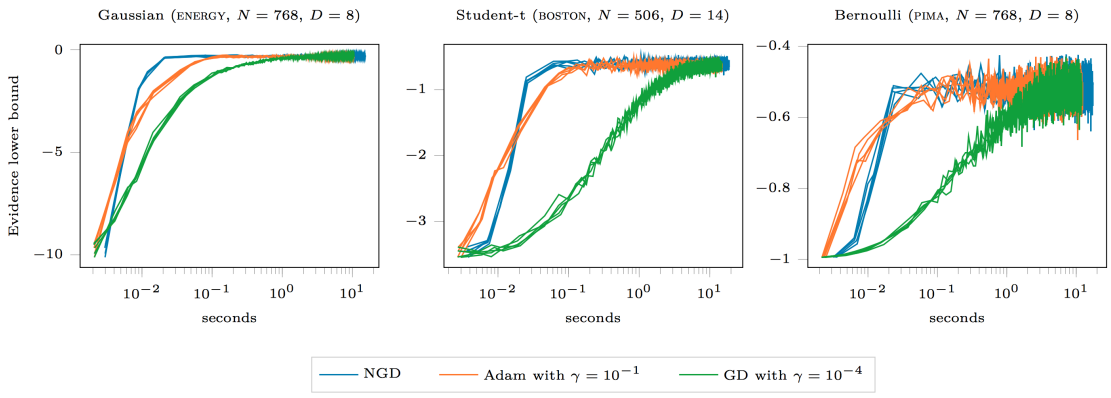


Figure 3.3: Stochastic optimization of the lower bound for fixed hyperparameters. The batch size is 256 and 5000 iterations are shown for five splits.

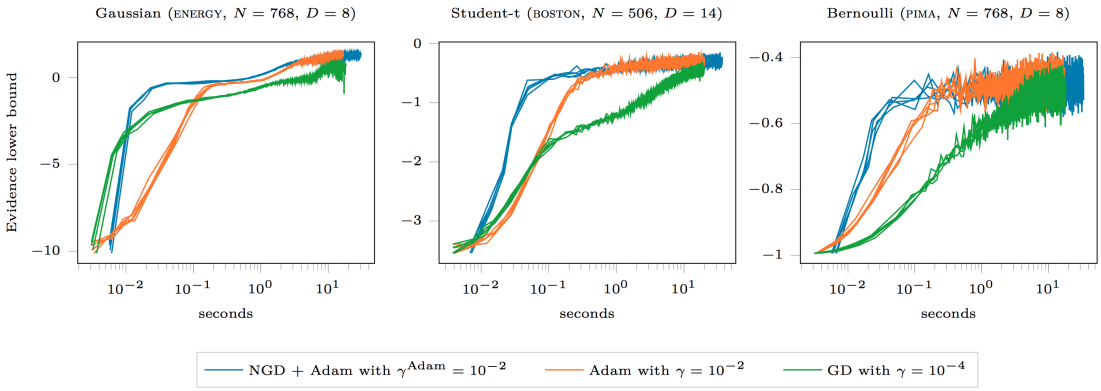


Figure 3.4: Joint optimization of the hyperparameters and the variational distribution. For natural gradients, a step of NGD on the variational parameters is alternated with a step of Adam on the hyperparameters. For Adam and GD the variational and hyperparameters are optimized together in a single objective. The batch size is 256 and 5000 iterations are shown for five splits.

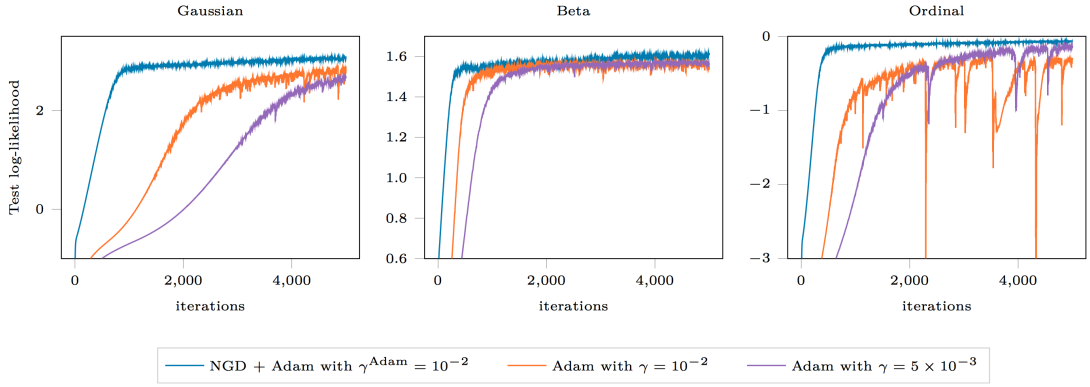


Figure 3.5: Optimization of the NAVAL dataset ($N = 11$K, $D = 16$), with three different likelihoods. The ill-conditioning of the variational distributions renders the optimization using ordinary gradients extremely difficult, even given a large number of iterations and different values for the Adam learning rate. The batch size is 256 and 5000 iterations are shown for a single split.

(ii) set $\gamma = \gamma_{\text{final}}$ for the remaining iterations. For the ENERGY, BOSTON and PIMA datasets we found that $\gamma_{\text{initial}} = 10^{-4}$, $\gamma_{\text{final}} = 10^{-1}$ and $K = 5$ were suitable values.

Fig. 3.3 shows the optimization of the ENERGY, BOSTON and PIMA datasets against wall-clock time with GD, Adam and NGD. We observe that NGD improves on Adam and GD after about 3×10^{-4} seconds (about 3 iterations). The advantages we see in the deterministic case appear to be realised in the stochastic setting.

3.4.3 Hyperparameters

An advantage of variational inference is that the ELBO can be optimized with respect to hyperparameters (we include also the inducing point inputs $\{\mathbf{z}_i\}_{i=1}^M$) as a proxy for the true marginal likelihood. Note that this is biased as the slack in the bound may depend on hyperparameter settings [Turner and Sahani, 2011]. Nevertheless, it has been found to work well in practice, so a prevalent approach is to optimize the hyperparameters and variational parameters together in a single objective. We cannot use natural gradients directly for the hyperparameters as we do not have a probability distribution for them. Instead, we use an alternating scheme where we perform a step of Adam on the hyperparameters (with step size γ^{Adam}), followed by a step of NGD on the variational parameters (with step size γ) We refer to this hybrid method as NGD+Adam and apply this approach to the same three datasets, using the same schedule for γ as before. We compare to optimizing the variational distribution and hyperparmeters in a single objective using Adam and GD.

Fig. 3.4 shows the results of stochastic optimization of the variational distribution together with hyperparameters. We see that NGD+Adam outperforms the other three methods in terms of wall-clock time.

3.4.4 When natural gradients are *essential*

In this section we present a practical situation where natural gradients are *essential*. The previous experiments demonstrated settings where all approaches could find the same solution in a reasonable time. This is not always the case, however, and we present a setting where the natural gradient approach can find a better solution than any method using ordinary gradient.

In ill-conditioned settings ordinary gradients suffer from instability [Sun et al., 2009] and slow convergence. As the natural gradient is invariant to parameterization, NGD is not adversely effected by issues of conditioning. We consider the commonly used NAVAL dataset, which has target values uniformly distributed in 51 increments between 0.95 and 1. We use this dataset with three different likelihoods: a Gaussian likelihood (rescaling the values to zero mean and unit variance), a single-parameter Beta likelihood[3] (rescaling to [0, 1]) and an ordinal likelihood (rescaling to $0, 1, \ldots, 50$) [Chu and Ghahramani, 2005], with bins uniformly spaced between -2 and 2. For NGD+Adam use a schedule with $\gamma_{\text{initial}} = 10^{-4}$, $\gamma_{\text{final}} = 10^{-1}$ and $K = 40$. We compare to the optimization of the lower bound with respect to hyperparameters and variational parameters using NGD+Adam and Adam.

Fig. 3.5 shows the optimization progress in terms of test log-likelihood after a large number of iterations. Note that ordinary gradient with Adam cannot achieve the optimal value, even after many iterations and with different step sizes.

3.4.5 Further likelihoods

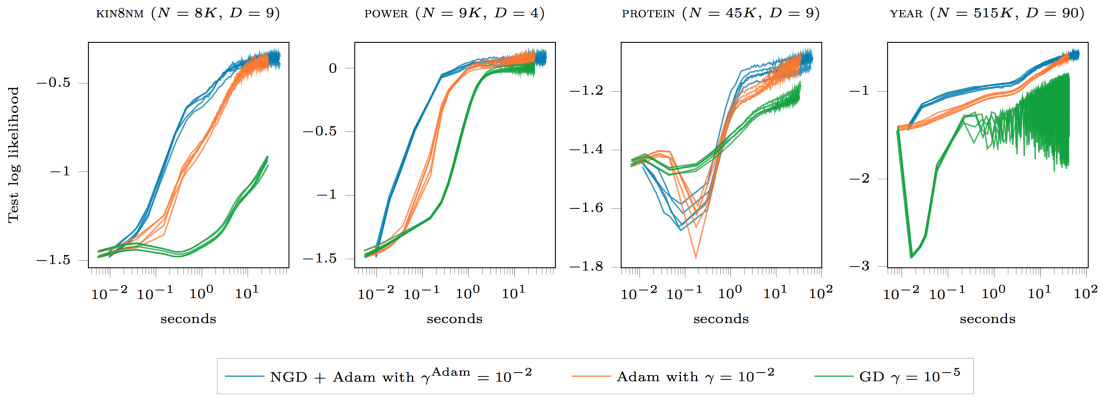


Figure 3.6: Optimization of hyperparameters and variational distributions for larger UCI datasets with a student-t likelihood.

In this section we apply NGD+Adam to 4 larger datsets from UCI corpus, using a student-t likelihood, and also the MNIST for multiclass classification. In both settings

[3]The usual α, β parameters are related by $\alpha = sm, \beta = s(1-m)$, with $m = \sigma(f)$ and $s > 0$ a hyperparameter.

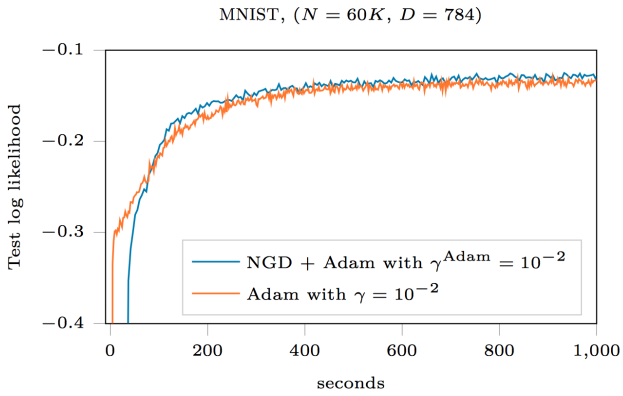


Figure 3.7: Optimization of the hyperparameters and variational distribution for the MNIST data, with a Robust-max multiclass likelihood. We see that after the first few initial iterations NGD+Adam outperforms Adam alone.

we find that natural gradients either find the optimal solution more quickly, or enable a solution to be found that cannot be obtained using ordinary gradients alone.

Fig. 3.6 shows the optimization in the UCI datasets with student-t likelihood, using a minibatch size of 256 and with the same schedule for γ as in the NAVAL experiment. For the KIN8NM, POWER and YEAR datasets we observe significant improvement over Adam.

Fig. 3.7 shows the result of MNIST multiclass classification using the standard train/test split and a batch size of 1024. The schedule for γ increases log-linearly from 10^{-6} to 0.02 over 2000 iterations. We see that the natural gradient approach outperforms Adam in terms of test loglikelihood.

3.4.6 The deep Gaussian process case

There are two sensible alternatives approach when applying the natural gradient method to the case of the deep GP. The first is to use natural gradients for just the final layer and the second is to use natural gradients for all layers. In the first case if the likelihood is Gaussian then quite large step sizes may be used robustly. If natural gradients are used for all layers then an adaptive step size may need to be used for the inner layer.

A significant complication in evaluating the effectiveness of the natural gradient is the potential non-convexity of the optimization problem. As both gradient descent and natural gradient descent are local methods there is no guarantee that a 'better' method in terms of progress per step will reach a better overall solution.

This issue is manifest also when optimizing hyperparameters in the single layer model, but there is an important distinction to be made. In the single layer case the optimization of the GP part is convex (for log-convex likelihoods), so a method that makes more progress per step is likely to be better overall. Solving the variational GP optimization perfectly (as in the 'collapsed' approach of Titsias [2009] in the Gaussian

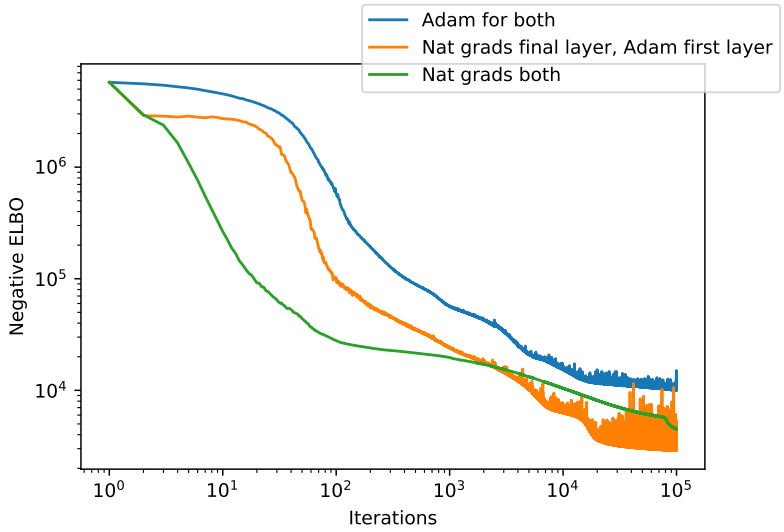


Figure 3.8: Optimization curves for a two layer DGP with fixed hyperparameters

likelihood case) will generally lead to better overall solutions for hyperparmeters, though examples could be constructed where this is not the case (for example, where the GP can be 'switched off' by setting the kernel variance to zero and modelling the data as noise). The final layer optimization is convex for log-convex likelihoods, so it is likely that natural gradients in the last layer will improve the overall performance. The optimization of the inner layer might well be mutlimodal, and so momentum based optimization could prove more effective at finding good global solutions, even though they might make less progress per step at the initial stages of optimization.

In practice, it seems the effects described in the previous paragraph are borne out in real examples. Natural gradients for the final layer appears highly advantageous, even with fixed step-size. Natural gradients for the inner layer requires some tuning, and tuning for good initial performance (in terms of progress per step) is not always as effective as using a momentum optimizer such as Adam for the inner layers. Figure 3.8 shows an example of these effects, using the same data from Chapter 2 and fixed hyperparameters. The `natural` parameterization is used whenever the natural gradients are applied, and the `mean/var-sqrt` otherwise. Note that using natural gradients for both layers gives a better bound for the first 2000 iterations, but after that the hybrid optimization of Adam for the inner layer and natural gradients for the final layer is more effective. Note also that using natural gradients for just the final layer is a strict (and significant) improvement over using Adam for both layers.

Using natural gradients for the final layer is particularly convenient in the case of a Gaussian likelihood as a quite large step size (for example 0.01) is feasible. While using natural gradients may have advantages in some situations it is not pursued further in this thesis.

3.5 Related work

The first use of natural gradients for variational inference goes back to Sato [2001], where it was shown that for an exponential family conditionally conjugate model (i.e., a model where classical fixed point variational updates can be derived in closed form), the NGD corresponds exactly to the fixed point variational update if $\gamma = 1$. This observation leads to an online version of the fixed point algorithm. This idea was made more explicit in [Hoffman et al., 2013], where it was termed stochastic variational inference and applied to a range of problems. The first example of natural gradients used in the non-conjugate case with Gaussian variational distribution can be found in [Honkela et al., 2010]. In this work, natural gradients are used for the variational mean. The inverse Fisher information for the mean has a particularly simple form (it is the precision), but the expression for the covariance is much more complicated and cumbersome to derive directly.

Natural evolution strategies (NES) [Sun et al., 2009] is closely related to variational inference. In NES a fitness function is optimized in expectation under a Gaussian. This converges to a zero entropy solution, so ordinary gradients cannot feasibly be used due to the problem of ill-conditioning. Natural gradients are therefore essential for a practical algorithm. In [Sun et al., 2009] the Fisher information for the `mean/var-sqrt` is calculated and inverted directly, which is inefficient. A similar result for the `mean/var` parameterization was presented in [Malagò and Pistone, 2015].

Recently, there have been several works employing natural gradients to approximations of non-conjugate components of a model. In the context of GPs, Khan et al. [2015, 2016] used a linearization of the non-conjugate terms and achieved impressive results. Johnson et al. [2016] use an auxiliary model to learn the approximate natural parameters with neural network likelihood, and then perform analytic updates on the conjugate approximation. Knowles and Minka [2011] use model-specific bounds to take approximate natural gradient steps in a variational message passing setting.

3.6 Discussion and conclusion

In all cases that we have investigated, we found that natural gradients accelerate convergence relative to methods using the ordinary gradient. In some cases the contrast is so severe that the ordinary gradient can require an unfeasibly large number of iterations to achieve the same results as the natural gradient. In practice, natural gradients are essential for finding a good solution in these situations. The drawbacks of the approach are that a schedule for γ must be specified. The success of the method relies on γ increasing to a reasonably large value (≈ 0.1) sufficiently quickly (< 1000 iterations). If γ needed to be kept small for much longer, then the advantage of the natural gradient method might be lost. Using a probabilistic line search [Mahsereci and Hennig, 2015] for NGD is a promising area for future research.

We have shown that natural gradients are useful for variational inference in non-conjugate sparse Gaussian process models. Natural gradients are particularly advantageous in problems where the ordinary gradient is crippled by the parameterization-dependent ill-conditioning. Such situations exist in practice. We have shown that natural gradients can be computed efficiently and with minimal effort using modern automatic differentiation techniques, and can be combined with modern optimizers such as Adam for hyperparameter learning. We compared six likelihoods and nine benchmark datasets, and found the natural gradient provided improvement in all cases.

Chapter 4

Gaussian Process Conditional Density Estimation

This chapter returns to the development of deep Gaussian process model, and considers the task of conditional density estimation (CDE) where the emphasis is on modelling non-Gaussian conditional distributions. As discussed in Chapter 1, a Gaussian process used directly is not interesting model for conditional density estimation as the marginals are always Gaussian. A model with non-Gaussian marginals can be constructed from a Gaussian process by introducing uncertainties in the inputs. If the inputs are independent Gaussian variables then this model is known as the Gaussian process latent variable model (GP-LVM) of Lawrence [2004]. A conditional variant was introduced in Wang and Neal [2012] and Damianou and Lawrence [2015]. This chapter develops this model to handle high dimensional inputs and outputs, and inference extends the approach of [Titsias and Lawrence, 2010a, Hensman et al., 2013]. Experiments are provided on a wide range tasks, showing the general applicability of the approach.

4.1 Introduction

The Gaussian process latent variable model (GP-LVM) [Lawrence, 2004] is a Gaussian process where the covariates are not observed but are instead given independent Gaussian priors. Typically the GP-LVM is used for settings with high dimensional outputs, where each output is an independent GP with shared covariates. Typically the outputs also share a covariance function and the likelihood is Gaussian, so the covariance is identical for each output. This model was original proposed as a

non-linear extension of PCA and was used for dimensionality reduction rather than density estimation.

A GP-LVM can be thought of as a DGP where the first layer has the white noise kernel, $k(x, x') = \delta_{xx'}$. The inducing point approach cannot be used for the first layer as the prior conditional contains no information: $p(f_*^1|\tilde{\mathbf{f}}^1) = p(f_*^1)$, where the notation is as in Chapter 2. The function values themselves must instead be represented in the variational distribution directly. This would pose a problem if the inputs were variables, but fortunately the inputs are fixed so there is no difficulty representing the function values.

The formulation of the GP-LVM as a DGP in this way is perhaps notationally confusing, as f^1 and f^2 have different properties and require different approaches for inference. Note that $f^1(x_n)$ does not depend on the *value* of x_n, but only on the *index*, i.e., $\text{cov}(f^1(x_i), f^1(x_j)) = \delta_{ij}$. Another consequence of the white noise process is that the variables that are not associated with data do not interact with the model, so do not need to be considered in the posterior. For this reason the $f^1(x_n)$ variables are referred to as 'latent variables', and for clarity a different letter w is used for the f^1 process, with $w_n = f^1(x_n)$. Note that it was argued in Chapter 2 that a definition of this nature is confusing as the w_n variable depends selectionally on its input, but that is not a problem here as the input is fixed as it is the first layer. This point is revisited in more detail in Chapter 5 in the context of a multi-layer model. Using the w notation for f^1 and defining f^2 as f, the GP-LVM can be written

$$p(y, f, w) = \mathcal{N}(y_n|f(w_n), \sigma^2) \tag{4.1}$$

with priors

$$f \sim \mathcal{GP}(m, k) \tag{4.2}$$
$$w_n \sim N(0, 1)\,. \tag{4.3}$$

It would be equivalent to write $w \sim \mathcal{GP}(0, k)$ where $k(x_i, x_j) = \delta_{ij}$, but there is no need to introduce the $w_* = w \setminus \{w_n\}$ to the model, as the w_* variables do no interact with any other variables and so posterior is the same as the prior.

The model of Wang and Neal [2012] is a hybrid of the Gaussian process regression and the GP-LVM. The model has the same priors as GP-LVM, but has likelihood

$$p(y, f, w) = \mathcal{N}(y_n|f([x_n, w_n])) \tag{4.4}$$

where $[\,\cdot\,,\,\cdot\,]$ denotes concatenation and x_n is an observed covariate for each y_n. Since w_n is a latent variable, which can be thought of as an additional covariate, this model was termed a 'Gaussian process with latent covariate' (GPLC) model. The idea of partially observed inputs appears in Damianou and Lawrence [2015], but the model has a different flavour as it is assumed that the covariates are missing for only some inputs. The GPLC model was used for 1D conditional density estimation in Wang and Neal [2012], with a sampling based inference.

If the GPLC model is interpreted as a DGP (that is, with f as f^2 and w_n as

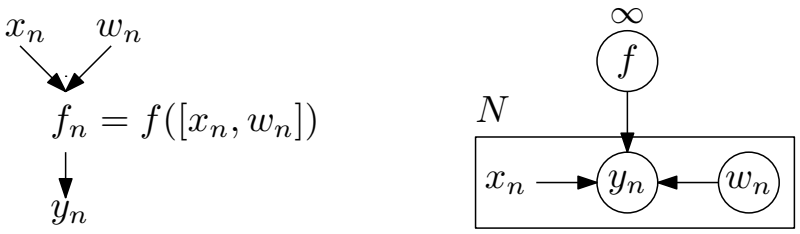
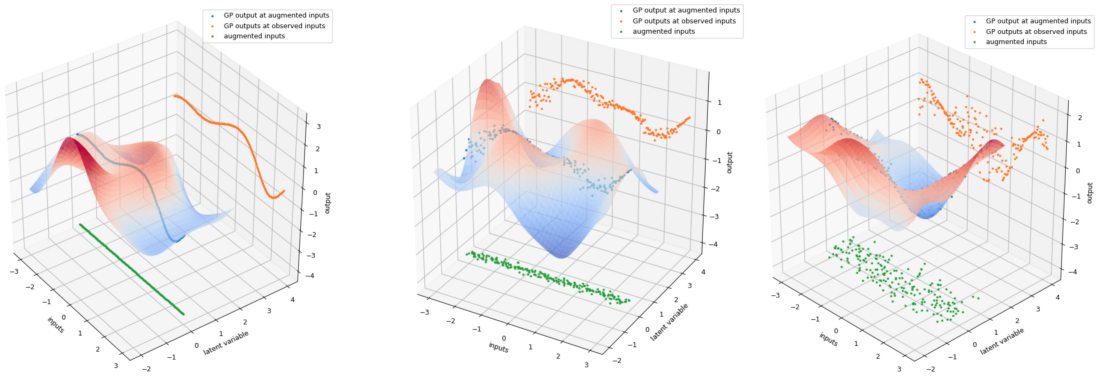


Figure 4.1: The GP-CDE model. Left: the likelihood evaluation diagram. Right: the graphical model.



(a) Zero variance (b) Small variance (c) Unit variance

Figure 4.2: Three draws from the prior, with different variances for the latent variable. The outputs (orange) are projected onto the back pane. The surface is a single draw from the two dimensional GP prior.

$w_n = f^1(x_n)$), then the graphical model is identical to the two layer DGP model from Chapter 2. As only the w variables associated with data are relevant, however, it is possible to notate only these variables in the graphical model. The graphical model under this interpretation is given in Figure 4.1, together with the likelihood evaluation diagram. Note that if two inputs happen to be identical $x_i = x_j$ for $i \neq j$ then it is still the case that $\text{cov}(w_i, w_j) = 0$.

To visualize the interaction of the latent variables with the inputs it necessary to plot in at least three dimensions. Figure 4.2 shows a plot of draws from the prior with three different values of the variance for the latent variable. Panel 4.2(a) shows the model with zero variance for the latent variables, which reduces to exactly the single layer model. Panel 4.2(b) shows a small variance latent variable and panel 4.2(c) shows the model described above with unit variance latent variables. Note that curvature in the latent variable direction leads to non-Gaussian marginals.

4.2 Inference

Exact inference is intractable in this model, so we use variational inference with a variational posterior $q(f, w) = q(f)q(w)$, where where $q(f)$ is the sparse GP posterior

from Chapter 1. It follows immediately that $q(w) = \prod_{i=1}^{N} q(w_i)$, and so the lower bound is

$$\mathcal{L}_q = \sum_{i=1}^{N} \left[\mathbb{E}_{q(f)q(w_i)} \log p\left(y_i | f([x_i, w_i])\right) - \mathrm{KL}[q(w_i)||p(w_i)]\right] - \mathrm{KL}[q(\tilde{\mathbf{f}})||p(\tilde{\mathbf{f}})] \quad (4.5)$$

We now consider two choices for $q(w_i)$: Gaussian and a free-form approach that is optimal subject to accurate numerical quadrature.

4.2.1 Gaussian variational distribution for the latent variables

The Gaussian approach $q(w_i) = \mathcal{N}(a_i, b_i^2)$ was first proposed by Titsias and Lawrence [2010a] in the GP-LVM model. A Gaussian $q(w_i)$ implies that the KL over the latent variables is closed-form, as both the prior and posterior are Gaussian. To calculate the log likelihood expectation Titsias and Lawrence [2010a] the expectation over f can be taken first, and the result is a quadratic form in $\mathbf{k}([x_i, w_i])$ (recall that $[\mathbf{k}([x_i, w_i])]_m = k([\tilde{x}_m, \tilde{w}_m], [x_i, w_i])$, where $\{[\tilde{x}_m, \tilde{w}_m]\}$ are the inducing points). For the RBF and polynomial kernels the moments under a Gaussian are available in closed form, so the expectation of the quadratic form can be computed analytically [Girard et al., 2003]. It is also possible to analytically optimize the parameters of $q(\tilde{\mathbf{f}})$ (in fact, the Gaussian form can be shown to be optimal [Titsias and Lawrence, 2010a]). The analytic results are expensive to compute, however, since they require the construction of a $N \times M \times M$ tensor. Instead, we resort to sampling from $q(w_i)$, using the reparameterization from Chapter 2. Viewing the model as a two layer DGP this approach is identical to that in Chapter 2, but with a differently parametrized $q(w)$.

In practice, rather than represent the Gaussian parameters a_i and b_i for each datapoint directly, we follow the technique popularized by the VAE literature [Kingma and Welling, 2013, Rezende et al., 2014] and instead amortize these parameters into a set of global parameters ϕ, where ϕ parameterizes an auxiliary function h_ϕ, (or 'recognition network') of the data: $a_i, b_i = h_\phi(x_n, y_n)$. This approach compresses the burden of optimizing N local parameters into optimizing a set of global parameters ϕ. The advantages are two fold: first, it avoids the linear parameter scaling so is computationally advantageous for large datasets, and second it is likely to result in a simpler optimization problem. To see this, consider training a model and then retaining on a new dataset with each of the points repeated twice. Under the direct approach the optimization of the new variational parameters needs to be done again from scratch, repeating the computation, but with the amortized approach the h_ϕ is already optimal for the new data and needs no further optimization.

4.2.2 Analytically optimal variational distribution for the latent variables

So far, we assumed that the variational distribution $q(w_i)$ is Gaussian. When $q(\,\cdot\,)$ is non-Gaussian, it is possible to integrate over w_n with quadrature as we detail in the following. We first bound the conditional $p(y|x,w)$ and use the same sparse variational posterior for the GP as before, to obtain

$$\log p(y|x,w) \geq \sum\nolimits_n \mathcal{L}_{w_n} - \mathrm{KL}[q(\tilde{\mathbf{f}})||p(\tilde{\mathbf{f}})]\,. \tag{4.6}$$

where $\mathcal{L}_{w_n} = \mathbb{E}_{f(x_n)} \log p(y_n|f([x_n,w_n]))$, which is available in closed form as the likelihood is Gaussian [Hensman et al., 2013]. By expressing the marginal likelihood as $p(y|x) = \int p(y|x,w)p(w)\mathrm{d}w$, we get

$$\log p(y|x) \geq \log \int \exp\left(\sum\nolimits_n \mathcal{L}_{w_n} - \mathrm{KL}[q(\tilde{\mathbf{f}})||p(\tilde{\mathbf{f}})]\right) p(w)\mathrm{d}w \tag{4.7}$$

$$= \log \sum\nolimits_n \int \exp\left(\mathcal{L}_{w_n}\right) p(w_n)\mathrm{d}w_n - \mathrm{KL}[q(\tilde{\mathbf{f}})||p(\tilde{\mathbf{f}})] \tag{4.8}$$

where we exploited the monotonicity of the logarithm to preserve the inequality. We can compute this integral with quadrature when w_n is low-dimensional (the dimensionality of x_n does not matter here). Assuming we have sufficient quadrature points, this gives the *analytically optimal* bound for $q(w_n)$. The analytical optimal approach does not resort to the Gaussian approximation for $q(w_n)$, so it is a tighter bound. The number of quadrature points necessary depends on the smoothness of $\exp(\mathcal{L}_{\mathcal{W}})$, but in practice we find 100 points is easily sufficient as the integrand in a smooth function.

Natural Gradient

Optimizing $q(\tilde{\mathbf{f}})$ together with $q(w)$ can be challenging due to problems of local optima and the strong coupling between the inducing outputs $\vec{u}_\ell$ and the latent variables $\mathcal{W}$. One option is to analytically optimize the bound with respect to the variational parameters of $\tilde{\mathbf{f}}$ following Titsias and Lawrence [2010a], but this prohibits the use of mini-batches and reduces the applicability to large-scale problems. Recall that the variational parameters of $q(\tilde{\mathbf{f}})$ are the mean and the covariance of the approximate posterior distribution $q(\tilde{\mathbf{f}}) = \mathcal{N}(\tilde{\mathbf{m}}, \tilde{\mathbf{S}})$ over the inducing outputs. We can use the natural gradient [Amari, 1998] approach from Chapter 3 to update the variational parameters $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$.

This approach has the attractive property of recovering exactly the analytically optimal solution for $q(\tilde{\mathbf{f}})$ in the full-batch case if the natural gradient step size is taken to be 1 and the number of $q(w_n)$ samples is taken to be large. In practice, the natural gradient is used for the Gaussian variational parameters $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$, and ordinary gradients are used for the inducing inputs $[\tilde{x}_m, \tilde{w}_m]$, the recognition network (if applicable) and other hyperparameters of the model (the kernel and likelihood parameters).

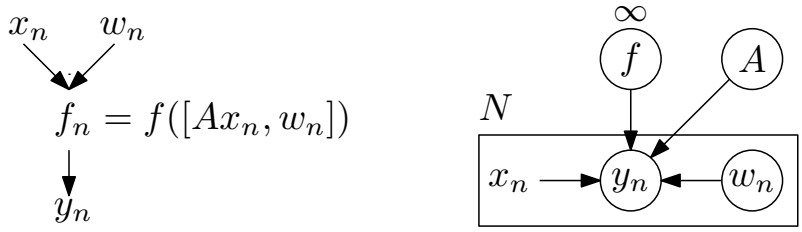


Figure 4.3: The GP-CDE model, with probabilistic input projections. Left: the likelihood evaluation diagram. Right: the graphical model.

4.2.3 Probabilistic linear transformations

Input For high-dimensional inputs it may not be appropriate to define a GP directly in the augmented space $[x_n, w_n] \in \mathbb{R}^{D_x+D_w}$. This might be the case if the input data is a one-hot encoding of many classes. We can extend our model with a linear projection to a lower-dimensional space before concatenating with the latent variables, $[Ax_n, w_n]$. We denote this projection matrix by A, as shown in Fig. 4.3.

We use an isotropic Gaussian prior for elements of A and a Gaussian variational posterior that factorizes between A and the other variables in the model: $q(f, w, A) = q(f)q(w)q(A)$. For Gaussian $q(w)$ the bound is identical as in (4.8) except that we include an additional $-\mathrm{KL}[q(A)||p(A)]$ term and include the mean and variance for Ax as the input of the GP. A similar approach was used in the regression case by Titsias and Lázaro-Gredilla [2013].

Output We can move beyond the assumption of a priori independent outputs to a correlated model by using a linear transformation of the outputs of the GP. This model is equivalent to a 'multi-output' GP model with a linear model of covariance between tasks. In the multi-output GP framework [Alvarez et al., 2012], the D_y outputs are stacked to a single vector of length ND_y, and a single GP is used jointly with a structured covariance. In the simplest case, the covariance can be structured as $R \otimes K$, where R can be any positive semi-definite matrix of size $D_y \times D_y$, and K is an $N \times N$ matrix. By transforming the outputs with the matrix $\mathcal{P}$ we recover exactly this model with $R = P^\top P$. Apart from the simplicity of implementation, another advantage is that we can handle degenerate cases (i.e., where the number of outputs is less than D_y) without having to deal with issues of ill-conditioning. It would be possible to use a Gaussian prior for P while retaining conjugacy, but in our experiments we use a non-probabilistic P and optimize it using ELBO.

4.3 Related work

The GP-CDE model is closely related to both supervised an unsupervised Gaussian process based models. If we drop the latent variables w our approach recovers standard multiple-output Gaussian process regression with sparse variational inference. If we drop the known inputs x and use only the latent variables, we obtain a Bayesian

GP-LVM Titsias and Lawrence [2010a]. Bayesian GP-LVMs are typically used for modeling complex distributions and non-linear mappings from a lower-dimensional variable into a high-dimensional space. By combining the GP-LVM framework with known inputs we create a model that outputs conditional samples in this high-dimensional space. Differently from Titsias and Lawrence 2010a, during inference we do not marginalize out the inducing variables $\tilde{\mathbf{f}}$ but rather treat them as variational parameters of the model. This scales our model to arbitrarily large datasets through the use of Stochastic Variational Inference (SVI) Hensman et al. [2013], Hoffman et al. [2013]. While the mini-batch extension to the Bayesian GP-LVM was suggested in Hensman et al. [2013], its notable absence from the literature may be due to the difficulty in the joint optimization of w and $\tilde{\mathbf{f}}$. We found that the natural gradients were essential to alleviate this problem. A comparison demonstrating this is presented in the experiments.

Wang and Neal [2012] proposed the Gaussian Process Latent Variable model (GP-LV), which is a special case of our model. The inference they employ is based on Metroplis sampling schemes and does not scale to large datasets or high dimensions. In this work, we extend their model using linear projection matrices on both input and output, and we present an alternative method of inference that scales to large datasets. Damianou and Lawrence [2015] also propose a special case of our model, though they use it for missing data imputation rather than to induce non-Gaussian densities. They also use sparse variational inference, but they analytically optimize $q(\tilde{\mathbf{f}})$ so cannot use mini-batches. Depeweg et al. [2016] propose a similar model, but use a Bayesian neural network instead of a GP.

The use of a recognition model, as in VAEs, was first proposed by Lawrence and Quiñonero-Candela [2006] in the context of a GP-LVM, though it was motivated as a constraint on the latent variable locations rather than an amortization of the optimization cost. Recognitions models were later used by Bui and Turner [2015] and by Dai et al. [2016] for deep GPs.

A GP model with latent variables and correlated multiple outputs was recently proposed in Dai et al. [2017]. In this model, the latent variables determine the correlations between outputs via a Kronecker-structured covariance, whereas we have a fixed between-output covariance. That is, in our model the covariance of the stacked outputs is $(PP^\top) \otimes (K_x K_w)$, whereas in Dai et al. [2017] the covariance is $K_w \otimes K_x$. These models are complementary and perform different functions. Bodin et al. [2017] proposed a model that is also similar to ours, but with categorical variables in the latent space. Other approaches to non-parametric density estimation include modeling the log density directly with a GP [Adams et al., 2009], and using an infinite generalization of the exponential family [Sriperumbudur et al., 2017] which was recently extended to the conditional case [Arbel and Gretton, 2018].

4.4 Experiments

Large-scale spatio-temporal density estimation We apply our model to a New York City taxi dataset to perform conditional spatial density estimation. The

Figure 4.4: Conditional densities (displayed as heat-maps: yellow means higher probability) of drop-off locations conditioned on the pick-up location (red cross).

dataset holds records of more than 1.4 million taxi trips, which we filter to include trips that start and end within the Manhattan area. Our objective is to predict spatial distributions of the drop-off location, based on the pick-up location, the day of the week, and the time of day. The two temporal features are encoded as sine and cosine with the natural periods, giving 6-dimensional inputs in total[1].

Table 4.1 compares the performance of 6 different models, unconditional and conditional Kernel Density Estimation (U-KDE, C-KDE), Mixture Density Networks (MDN-k, $k = 1, 5, 10, 50$) [Bishop, 1994], our GP-CDE model, a simple GP model and the unconditional GP-LVM [Titsias and Lawrence, 2010a]. We evaluate the models using negative log predictive probability (NLPP) of the test set. The test sets are constructed by sequentially adding points that have greatest minimum distance from the testing set. In this way we cover as much of the input space as possible. We use a test set of 1000 points, and vary the number of training points to establish the utility of models in both sparse and dense data regimes. We use 1K, 5K and 1M randomly selected training points to evaluate the models in both sparse and dense data regimes.

Unconditional KDE (U-KDE) ignores the input conditions. It directly models the drop-off locations using Gaussian kernel smoothing. The kernel width is selected with cross-validation. The Conditional KDE (C-KDE) model uses the 50 nearest neighbors in the training data, with kernel width taken from the unconditional model. The table shows that the conditional KDE model performs better than the unconditional KDE model for all conditions. This suggests that the conditioning on the pick-up location and time strongly effects the drop-off location. If the effect of conditioning were slight, the unconditional model should perform better as it has access to all the data.

We also evaluate several Mixture Density Networks (MDNs) models with differing

[1]See `https://github.com/hughsalimbeni/bayesian_benchmarks` for the data.

Table 4.1: NLPP for Manhattan data (lower is better). The models are trained on different dataset sizes.

	1K	5K	1M
GP-LVM	2.61	2.52	2.43
GP	2.68	2.67	2.67
GP-CDE	**2.31**	**2.22**	2.13
U-KDE	2.49	2.5	2.35
C-KDE	2.40	2.38	2.314
MDN-1	2.83	2.77	2.65
MDN-5	2.72	2.55	2.16
MDN-10	3.17	2.66	2.06
MDN-50	5.09	3.08	**1.97**

number of mixing components (MDN-k, where k is the number of components), using fully connected neural networks with 3 layers. The MDN model perform poorly except in the large data regime, where the model with the largest number of components is the best performing. The MDN with a large number of components can put mass at localized locations, which for this data is likely to be appropriate as the taxis are confined to streets.

We test three GP-based models: our GP-CDE model with 2-dimensional latent variables, and two special cases: one without conditioning (GP-LVM) and one without latent variables (GP). The GP-LVM [Titsias and Lawrence, 2010a] is our model without the conditioning, and does not perform well on this task as it has not access to the inputs and models all conditions identically. The GP model has no latent variables and independent Gaussian marginals, and so cannot model this data well as the drop-off location is quite strongly non-Gaussian.

The GP-CDE performs best on this dataset for the small data regimes. For the large data case the MDN model is superior. We attribute this to the high density of data when 1 million training points are used. We used a 2D latent space and Gaussian $q(\mathcal{W})$ for the latent variables, with a recognition network amortizing the inference. We use the RBF kernel and use Monte Carlo sampling to evaluate the bound. For training we use the Adam optimizer with a exponentially decaying learning rate starting at 0.01 for the hyperparameters, the inducing inputs and on the recognition network parameters. Natural gradient steps of size 0.05 are used for the GP's variational parameters. Fig. 4.4 shows the density of our GP-CDE model for two different conditions.

Few-shot learning We demonstrate the GP-CDE model for the challenging task of few-shot conditional density estimation on the omniglot dataset. Our task is to obtain a density over the pixels given the class label. We use the training/test split from Lake et al. [2015], using all the examples in the training classes and four samples from each of the test classes. The inputs are one-hot encoded (1623 classes)

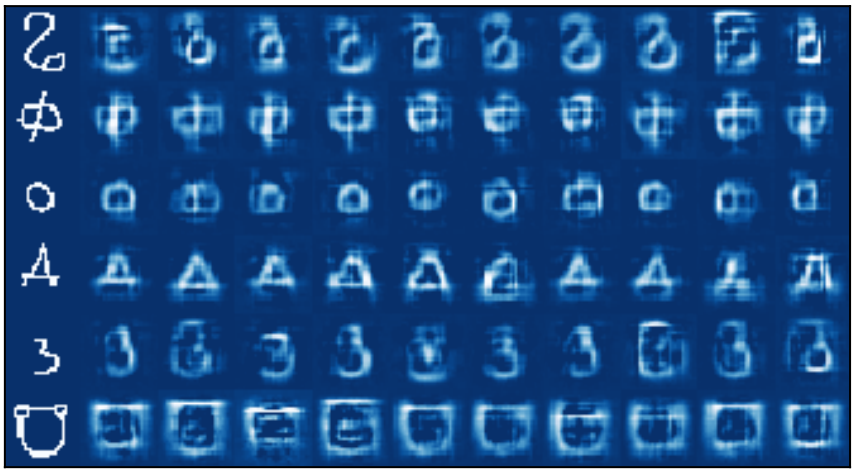


Figure 4.5: Sample images for 4-shot learning. Left column is a true (unseen) image, remaining columns are samples from the posterior conditioned on the same label.

and the outputs are the pixel intensities, which we resize to 28×28.

We apply a linear transformation on both the input and output (see Section 4.2.3). We use a 1623×30 matrix $\mathbf{A}$ with independent standard-normal priors on each component to project the labels onto a 30-dimensional space. To prevent the model from overfitting, it is important to treat the $\mathbf{A}$ transformation in a Bayesian way and marginalize it.

To correlate the outputs a priori we use a linear transformation $\mathcal{P}$ of the GP outputs, which is equivalent to considering multiple outputs jointly in a single GP with a Kronecker-structured covariance. See Section 4.2.3. We use 400 GP outputs so $\mathcal{P}$ has shape 400×784. To initialize $\mathcal{P}$ we use a model of local correlation by using the Matérn-$\frac{5}{2}$ kernel with unit lengthscale on the pixel coordinates and taking the first 400 eigenvectors scaled by the square-root eigenvalues. We then optimize the matrix $\mathcal{P}$ as a hyperparameter of the model. Learning the $\mathcal{P}$ matrix is a form of transfer learning: we update our prior in light of the training classes to make better inference about the few-shot test classes.

We obtain a log-likelihood of 7.2×10^{-2} nat/pixel, averaging over the all the test images (659 classes with 16 images per class). We train for 25K iterations with the same training procedure as in the previous experiment. Samples from the posterior on a selection of test classes are shown in Fig. 4.5.

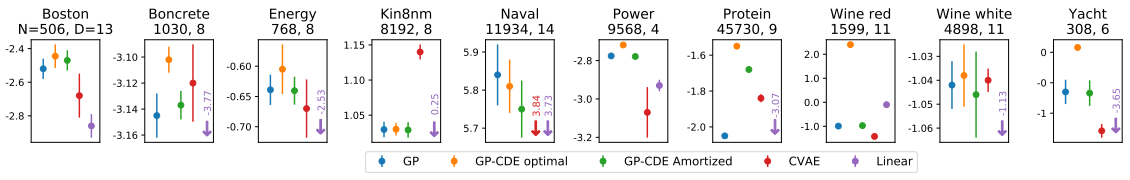


Figure 4.6: Test log-likelihood of the GP, the optimal GP-CDE, the amortized GP-CDE, the CVAE, and a Linear model on 10 UCI datasets. Hihgher is better.

Heteroscedastic noise modeling We use 10 UCI regression datasets to compare two variants of our CDE model with a standard sparse GP and a CVAE. Since we model a 1D target we consider w_n to be uni-dimensional, allowing us to use the quadrature method (Section 4.2.2) to obtain the bound for an analytically optimal $q(w_n)$. We compare also to an amortized Gaussian approximation to $q(w_n)$, where we use a three-layer fully connected neural network with tanh activations for the recognition model. In all three models we use a RBF kernel and 100 inducing points, optimizing for $20K$ iterations using Adam optimizer for the hyperparameters and a natural gradient optimizer with step size 0.1 for the Gaussian variational parameters. The quadrature model use Gauss-Hermite quadrature with 100 points. For the CVAE we use, given the modest size of the UCI datasets, a relatively small encoder and decoder network architecture together with dropout.

Fig. 4.6 shows the test log-likelihoods using 20-fold cross validation with 10% test splits. We normalize the inputs to have zero mean and unit variance. We see that the quadrature CDE model outperforms the standard GP and CVAE on many of the datasets. The optimal GP-CDE model performs better than the GP-CDE with Gaussian $q(w)$ on all datasets. This can be attributed to three reasons: we impose fewer restrictions on the variational posterior, there is no amortization gap (i.e. the recognition network might not find the optimal parameters [Cremer et al., 2018]), and problems of local optima are likely to be less severe as there are fewer variational parameters to optimize.

Density estimation of image data In this experiment we compare the test log-likelihood of the GP-CDE and the CVAE Sohn et al. [2015a] on the MNIST dataset for the image generation task. We train the models with $N = 2, 4, 8, \ldots, 512$ images per class, to test their utility in different data regimes. The model's input is a one-hot encoding of the image label which we concatenate with a 2-dimensional latent variable. We use all $10,000$ test images to calculate the average test log-likelihood, which we estimate using Monte Carlo.

For the CVAE's encoder and decoder network architecture we follow Wu et al. [2016a] and regularize the network using dropout. The GP-CDE has the same setup as in the few-shot learning experiment, except that we set the shape of the output mixing matrix $\mathcal{P}$ to 50×784. We reduce the size of $\mathcal{P}$, compared to the omniglot experiment, as the MNIST digits are relatively easier to model. Since we are considering small datasets in this experiment the role of the mixing matrix becomes more important: it enables the encoding of prior knowledge about the structure in images.

Wu et al. [2016a] point out that when evaluating test-densities for generative models, the assumed noise variance σ^2 plays an important role, so for both models we compare two different cases: one with the likelihood variance parameter fixed and one where it is optimized. Table 4.2 shows that in low-data regimes the highly parametrized CVAE severely overfits to the data and underestimates the variance. The GP-CDE operates much more gracefully in these regimes: it estimates the variance correctly, even for $N = 2$ (where N is the number of *training* points), and the gap between train/test log-likelihood is considerably smaller.

	CVAE: Fixed σ^2		CVAE: σ^2 optimized			GP-CDE: Fixed σ^2		GP-CDE: σ^2 optimized		
N	Test	Train	Test	Train	σ^2_{opt}	Test	Train	Test	Train	σ^2_{opt}
2	-129.72	180.97	**-1296.63**	956.39	**0.01378**	161.9	242.2	**74.01**	**130.4**	**0.0303**
4	-60.03	178.22	-759.18	956.26	0.01364	195.2	254.2	86.59	160.3	0.0310
256	52.17	76.18	218.08	325.72	0.03272	606.2	545	108.1	105.4	0.0378
512	54.48	65.30	244.88	286.38	**0.03407**	606.7	512	124.2	120.7	0.0388

Table 4.2: Log-likelihoods of the CVAE and GP-CDE models. N is the number of images per class. Higher test log-likelihood is better.

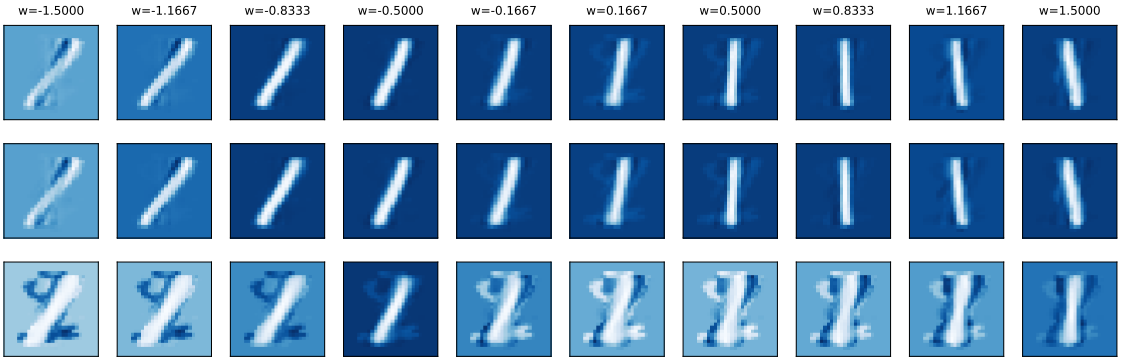


Figure 4.7: The mean of the GP mapping conditioned on the 1D latent variable, for a dataset of 100 '1's. Top: the analytic solution of Titsias and Lawrence [2010a]. Middle: our natural gradient approach, with a step size of 0.1. Bottom: using the Adam optimizer to optimize the variational parameters $\mathbf{m}$ and $\mathbf{S}$. We see that the ordinary gradient approach is prone to getting 'stuck' in poor local optima, due to the difficulty of optimization.

Necessity of natural gradients Natural gradients are a vital component of our approach. We demonstrate this with the simplest possible example modeling a dataset of 100 '1' digits, using an unconditional model with no projection matrices, no mini-batches and no recognition model (i.e. exactly the GP-LVM in Titsias and Lawrence [2010a]). We compare our natural gradient approach with step size of 0.1 against using the Adam optimizer (learning rate 0.001) directly for the variational parameters. We compare also to the analytic solution in Titsias and Lawrence [2010a], which is possible as we are not using mini-batches. We find that the analytic model and our natural gradient method obtain test log-likelihoods (using all the '1's in the standard testing set) of 1.02, but the ordinary gradient approach attains a test log-likelihood of only -0.13. See Fig. 4.7 for samples from the latent space, and Fig. 4.8 for the training curves. We see that the ordinary gradient model cannot find a good solution, even in a large number of iterations, but the natural gradient model performs similarly to the analytic case.

A note on attribution: the taxi and MNIST experiments were primarily implemented by collaborator Vincent Dutordoir, but are included for completeness. Vincent also implemented the CVAE baselines. I implemented the omniglot, UCI and natural gradients experiments.

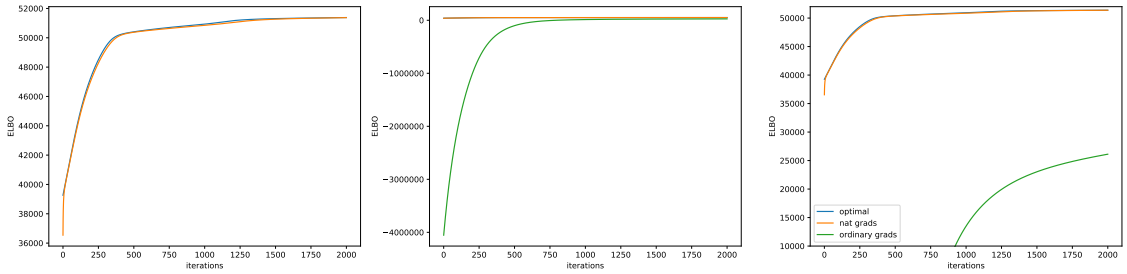


Figure 4.8: The training objective for the dataset of 100 '1's. The three plots show the same three curves with different ranges on the y-axis to highlight the similarities and differences. We can see that natural gradients provide a striking improvement. See Fig. 4.7 for the samples at the end of training

4.5 Conclusion

We presented a model for conditional density estimation with Gaussian processes. Our approach extends prior work in three ways. We perform Bayesian linear transformations on both input and output spaces to allow for the modeling of high-dimensional inputs and strongly-coupled outputs. Our model is able to operate in low and high data regimes. Compared with other approaches we have shown that our model does not over-concentrate its density, even with very few data. For inference, we derived an optimal posterior for the latent variable inputs and we demonstrated the usefulness of natural gradients for mini-batched training of GPs with uncertain inputs. These improvements provide us with a more accurate variational approximation, and allow us to scale to larger datasets than were previous possible. We applied the model in different settings across a wide range of dataset sizes and input/output domains, demonstrating its general utility.

Chapter 5

Deep Gaussian Processes with Importance-Weighted Variational Inference

As an example, consider a 1D density estimation task formed from the letters '*DGP*', where the inputs are the horizontal coordinates and the outputs are the vertical coordinates of points uniformly in the glyphs. The marginals are multimodal and piece-wise constant, and also change non-smoothly in some regions of the space (for example changing discontinuously from a bi- to tri-modal density mid-way through the 'G'). Figure 5.1 shows the posterior of a two-layer DGP with this data. Posteriors from other models (together with the training data) are shown in Figure 5.4.

This chapter extends the previous chapter to the deep case, and introduces a new form of importance-weighted variational inference for the latent variables. The approach revisits the original construction of Damianou and Lawrence [2013a], which includes variables that are a priori independent for each data point. Independent variables are important for modelling non-Gaussian marginals, as otherwise function values with the same input will be perfectly correlated. Differently from Damianou and Lawrence [2013a], we introduce the independent variables to the model as latent covariates (that is, as additional inputs to the GP) [Wang and Neal, 2012] rather than additively as process noise. As we have a clean separation between variables that are correlated and variables that are independent, we can apply appropriate inference for each. Similarly to the previous chapter, for the correlated variables we use a sparse variational GP to represent the posterior [Titsias, 2009, Matthews et al., 2016a]. For the uncorrelated variables we use a mean-field approach. We first derive a straightforward combination of the stochastic variational approach of Chapter 2 with a mean-field Gaussian approximation for the latent variables. We then propose a novel importance-weighted scheme that improves upon the Gaussian approach and

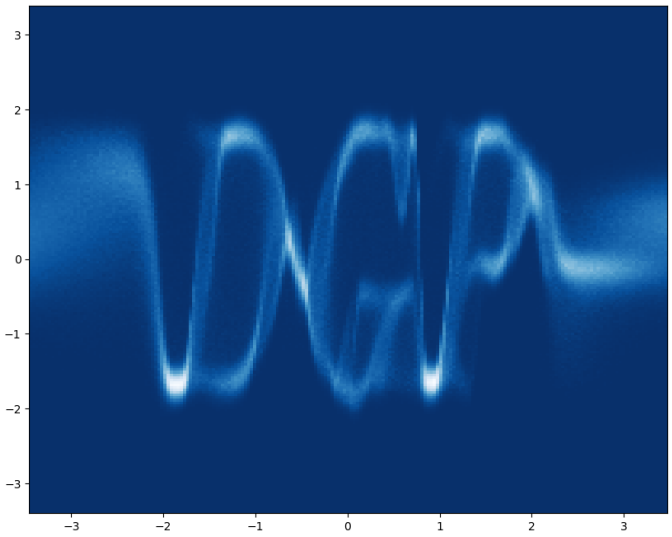


Figure 5.1: Posterior density from a two-layer model, illustrating non-Gaussian marginals and non-smooth input dependence.

trades additional computation for accuracy while retaining analytic results for the GP parts.

Our results show that the deep Gaussian process with latent variables is an effective model for real data. We investigate a large number of datasets and demonstrate that highly non-Gaussian marginals occur in practice, and that they are not well modelled by the noise-free approach of Salimbeni and Deisenroth [2017b], nor by the single layer approach of Chapter 4. We also show that our importance-weighted scheme is always an improvement over variational inference, especially for the deeper models.

5.1 Model

Our DGP model is built from two components (or *layers*) that can be freely combined to create a family of models. The two types of layers have orthogonal uses: one is for modelling *epistemic* uncertainty (also known as *model* or *reducible* uncertainty, when the output depends deterministically on the input, but there is uncertainty due to lack of observations) and the other is for modelling *aleatoric* uncertainty (also known as *irreducible* uncertainty, when the output is inherently random). Both layers are Gaussian processes, but we use the term *latent variable* when the process has the white noise covariance, and we use *Gaussian process* when the covariance has no noise component. Each layer takes an input and returns an output. The model is constructed by applying the layers sequentially to the input x_n to get a density over an output y_n.

5.1.1 Gaussian process (GP) layer

The Gaussian process layer defines a set of continuously indexed random variables, which are a priori jointly Gaussian, with mean and covariance that depend on the indices. We write f for the full set of variables (or 'function') and $f(x)$ the particular variable with index x. The notation $f \sim \mathcal{GP}(\mu, k)$ states that for any finite set $\{x_i\}$, the variables $\{f(x_i)\}$ are jointly Gaussian, with $\mathbb{E}\left(f(x_i)\right) = \mu(x_i)$ and $\mathrm{cov}\left(f(x_i), f(x_j)\right) = k(x_i, x_j)$, where μ is a mean function and k is a positive semi-definite covariance function. In this work, we always use 'noise free' kernels, which satisfy $k(x, x) = \lim_{x' \to x} k(x, x')$.

The GP prior is defined for all inputs, not just the ones associated with data. When we notate the GP function in a graphical model (see Figure 5.2), all the variables appear together as f. We include an infinity symbol in the graphical model to indicate that the GP node represents an infinite collection of random variables, one for every possible input.

5.1.2 Latent-variable (LV) layer

The latent-variable layer introduces variables to the model that are independent for each data point. As we wish to interpret these variables as unobserved covariates we introduce them through concatenation rather than through addition, as in previous work [see, for example Dai et al., 2016, **?**, Damianou and Lawrence, 2013a]. We use the notation $[x_n, w_n]$ to denote the concatenation of x_n with w_n. Throughout this work each component of w_n will be distributed as $\mathcal{N}(0, 1)$. For input x_n, the output of the latent variable layer is $[x_n, w_n]$.

5.1.3 Example model: LV-GP-GP

For the purpose of demonstrating the details of inference in the next section, we focus on a particular model. The model has an initial layer of latent variables followed by two Gaussian process layers. We refer to this model as 'LV-GP-GP'. The graphical model indicating the a priori statistical relationship between the variables is shown in Figure 5.2. The likelihood evaluation diagram is also shown. The figure also shows the GP-LV-GP model for comparison, which shares a graphical model with LV-GP-GP but differs in the evaluation diagram. Writing y and w for $\{y_n\}$ and $\{w_n\}$, the likelihood of the LV-GP-GP model is $p(y|f, g, w) = \prod_n p(y_n|f, g, w_n)$, with

$$p(y_n|f, g, w_n) = \mathcal{N}(y_n|f(g([x_n, w_n])), \sigma^2)\,. \tag{5.1}$$

The priors are

$$w_n \sim \mathcal{N}(0, 1)\,,\ f \sim \mathcal{GP}(\mu_1, k_1)\,,\ g \sim \mathcal{GP}(\mu_2, k_2)\,.$$

Both models are appropriate for continuous-valued data, and can model heavy and asymmetric tails, as well as multimodality, as we later demonstrate. We discuss choices for the mean and covariance functions in section 5.1.6.

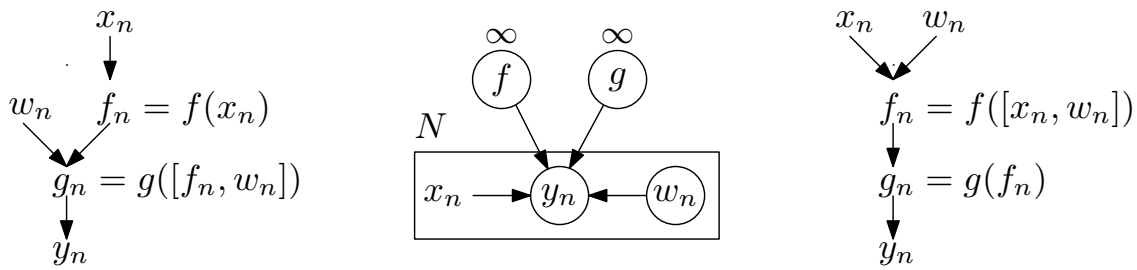
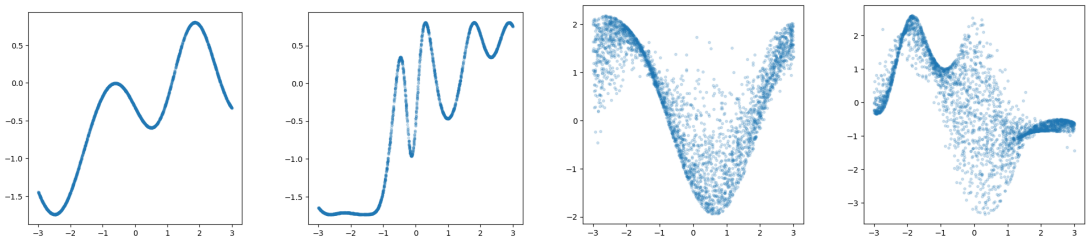


Figure 5.2: The DGP with latent variables in two configurations: GP-LV-GP (left) and LV-GP-GP (right). The graphical model (centre) is the same for each.



(a) GP (b) GP-GP (c) LV-GP (d) LV-GP-GP

Figure 5.3: A single sample from the prior for 4 illustrative models. The GP and GP-GP model have no latent variables. Their samples appear as deterministic functions.

5.1.4 Model variants

The LV-GP-GP model can be extended by adding more layers, and by adding more latent variables. For example, we could insert an additional GP layer,

$$p(y|f,g,h,w) = \prod_n \mathcal{N}\left(y_n|h(f\left(g([x_n,w_n])\right)),\sigma^2\right) ,$$

where $h \sim \mathcal{GP}(\mu_3, k_3)$. We refer to this model as LV-GP-GP-GP. Alternatively we could place the latent variables between the GPs, and have instead $f([g(x_n), w_n])$ for the conditional mean. We refer to this model as GP-LV-GP. Other models are analogously defined. For example, GP-GP is as above but with no latent variable. We refer to models without latent variables as 'noise-free'.

Prior samples from models with one and two GP layers are shown in Figure 5.3 to illustrate their properties. All the GP layers have 1D outputs and use the RBF kernel with unit lengthscale. For the latent variable models, the number of GPs above the latent variable layer determines the complexity of the marginals, but the number of GPs in total determines the complexity of the functional mapping.

5.1.5 Multiple outputs

All variables in our model (including the model inputs and outputs) may be vector valued. The 'multiple output GP' is a GP with a covariance function defined between outputs $k_{dd'}(x, x')$, where d indexes output dimension and x indexes the input. In

this notation, the independent output approach of Damianou and Lawrence [2013a] can be written as $k_{dd'}(x, x') = \delta_{dd'} k(x, x')$. For all our models, we consider a linear model of covariance between outputs: $k_{dd'}(x, x') = \sum_e P_{de} k_e(x, x') P_{ed'}$ [Alvarez et al., 2012]. For $k_e(x, x') = k(x, x')$ this is equivalent to assuming a Kronecker structured covariance between the outputs stacked into a single vector (i.e. with covariance $(P^\top P) \otimes K$), and is also equivalent to multiplying independent GPs stacked into a vector by the matrix P.

5.1.6 Mean and covariance functions

The DGP model suffers from a pathology if used with zero mean GPs at each layer [Duvenaud et al., 2014, Dunlop et al., 2017]. To remedy this problem, Duvenaud et al. [2014] propose concatenating the inputs with the outputs at each GP layer, whereas Salimbeni and Deisenroth [2017b] use a linear mean function to address this issue. We follow the latter approach. The inference we present in the next section is agnostic to covariance function, so we are free to make any choice. In our experiments, we use an RBF kernel for each layer, sharing the kernel over the different outputs.

5.2 Inference in the LV-GP-GP model

In this section, we present two approaches for approximate inference: the first with variational inference and a second with importance-weighted variational inference. Both schemes can optionally amortize the optimization of the local variational parameters (often referred to as 'auto-encoding'), are scalable through data subsampling, and can exploit natural gradients of the variational parameters for the final layer. While the variational approach is a straightforward extension of the doubly-stochastic method by Salimbeni and Deisenroth [2017b], the importance-weighted approach is more subtle and requires a careful choice of variational distribution to retain the analytic results for the final layer.

5.2.1 Variational inference

Variational inference seeks an approximate posterior that is close to the true posterior in terms of KL divergence. The posterior is typically restricted to some tractable family of distributions, and an optimization problem is formed by minimizing the KL divergence from an approximate posterior to the true posterior. Equivalently, the same objective can be obtained by applying Jensen's inequality to an importance-weighted expression for the marginal likelihood [Domke and Sheldon, 2018]. For an approximate posterior, we follow Damianou and Lawrence [2013a] and use a mean-field Gaussian distribution for the latent variables $q(w) = \prod_n q(w_n)$ with $q(w_n) = \mathcal{N}(a_n, b_n)$, and independent processes for the functions. The posterior

density[1] then has the same structure as the prior: $q(w, f, g) = q(w)q(f)q(g)$. We begin by writing the (exact) marginal likelihood as

$$p(y) = \mathop{\mathbb{E}}_{f,g,w}\left[p(y|f, g, w)\frac{p(f)p(g)p(w)}{q(f)q(g)q(w)}\right], \tag{5.2}$$

where the expectations are taken with respect to the variational distributions. Applying Jensen's inequality to the logarithm of both sides of (5.2), we obtain

$$\log p(y) \geq \sum_n (A_n - \mathrm{KL}_{w_n}) - \mathrm{KL}_f - \mathrm{KL}_g\,, \tag{5.3}$$

where we have used the short-hand KL_h for $\mathrm{KL}(q(h)||p(h))$, and A_n is given by

$$A_n = \mathbb{E}_{f,g,w_n} \log p(y_n|f, g, w_n)\,. \tag{5.4}$$

By considering a variational distribution over the entire function for f and g we avoid the difficulty of representing the indeterminately indexed inner-layer variables. We require only that KL_f and KL_g are finite, which is possible if we construct $q(f)$ and $q(g)$ as measures with respect to their priors. We follow the technique described in previous chapters and form these posteriors by conditioning the prior on some finite set of inducing points with a parameterized Gaussian distribution. As before, the priors and variational posteriors over f and g are independent (the coupling is only in the likelihood), so the results from the single layer GP apply to each layer.

To evaluate A_n exactly is intractable except in the single-layer case (and then only with a certain kernel; see Titsias and Lawrence [2010a] for details), so we rely on a Monte Carlo estimate. To obtain unbiased samples we first reparameterize the expectations over w_n and g, where $w_n = a_n + \epsilon_1\sqrt{b_n}$ and $g([x_n, w_n]) = \mu_2([x_n, w_n]) + \epsilon_2\sqrt{k_2([x_n, w_n], [x_n, w_n])}$ where $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 1)$. We then obtain an estimate of A_n by sampling from ϵ_1, ϵ_2, which is the 'reparameterization trick' popularized by Rezende et al. [2014], Kingma and Welling [2013]. After these two expectations have been approximated with a Monte Carlo estimate we can take the expectation over f analytically as the likelihood is Gaussian.

5.2.2 Importance-weighted variational inference

Jensen's inequality is tighter if the quantity inside the expectation is concentrated about its mean [Domke and Sheldon, 2018]. To decrease the variance inside (5.2) while preserving the value, we can replace the w term with a sample average of K terms,

$$p(y) = \mathbb{E}_{f,g,w}\frac{1}{K}\sum_{k=1}^{K} p(y|f, g, w^{(k)})\frac{p(w^{(k)})}{q(w^{(k)})}\frac{p(f)p(g)}{q(f)q(g)}\,. \tag{5.5}$$

[1] We abuse notation and write a process as if it has a density. The derivation can be made rigorous as the densities only appear as expectations of ratios. See Matthews et al. [2016a] for details.

The expression inside the expectation in (5.5) is known as the importance sampled estimate (with respect to w), motivating the term 'importance-weighted variational inference' when we take the logarithm of both sides and form a lower bound using Jensen's inequality. Applying Jensen's inequality to (5.5), we have the lower bound

$$\log p(y) \geq \sum_{n=1}^{N} B_n - \mathrm{KL}_f - \mathrm{KL}_g \,, \tag{5.6}$$

where the data-fit term B_n is given by

$$B_n = \mathbb{E}_{f,g,w_n} \log \frac{1}{K} \sum_k p(y_n|f,g,w_n^{(k)}) \frac{p(w_n^{(k)})}{q(w_n^{(k)})} \,. \tag{5.7}$$

This approach provides a strictly tighter bound [Burda et al., 2015, Domke and Sheldon, 2018], but the expression for B_n is less convenient to estimate than A_n as we cannot apply the analytic result for the f expectation due to the non-linearity of the logarithm. We must therefore resort to sampling for the expectation over f as well as g and w_n, incurring potentially high variance. This seems like an unacceptable price to pay as we have not gained any additional flexibility over f. In the next section, we show how we can retain the analytic expectation for f while exploiting importance-weighted sampling for w.

5.2.3 Analytic final layer

The expression in (5.7) does not exploit the conjugacy of the Gaussian likelihood with the final layer. In this section, we present a novel two-stage approach to obtain a bound that has all the analytic properties of the variational bound (5.3), but with improved accuracy. Our aim is to obtain a modified version of (5.7) but with the f expectation taken over the logarithm of the likelihood, since this expectation is tractable. We will show that we can find a bound that replaces the term $p(y_n|f,g,w_n)$ in (5.7) with $\exp \mathbb{E}_f \log p(y_n|f,g,w_n)$, which we can compute analytically. It is worth noting that since the likelihood is Gaussian we could analytically integrate out f to obtain $p(y|g,w)$, though doing so precludes the use of minibatches and incurs $\mathcal{O}(N^3)$ complexity. It is not surprising, then, that it is possible to 'collapse' the bound over f approximately. Our approach is inspired by the partially collapsed approaches in Hensman et al. [2012]. We begin by applying Jensen's inequality to the f expectation in (5.2):

$$\log p(y|g,w) \geq \sum_n L_n(g,w_n) - \mathrm{KL}_f,$$

where the data term is given by

$$L_n(g,w_n) = \mathbb{E}_f \log p(y_n|f,g,w_n) \,. \tag{5.8}$$

The expression for $L_n(g,w_n)$ is available in closed form as the conditional likelihood is Gaussian [see, for example, Titsias, 2009] . Applying the exponential function to

both sides of (5.8) gives the bound

$$p(y|g,w) \geq \exp\left[\sum_n L_n(g,w_n) - \mathrm{KL}_f\right]. \tag{5.9}$$

Returning again to (5.2), the exact marginal likelihood can be expressed equivalently with f marginalized as

$$p(y) = \mathop{\mathbb{E}}_{g,w} p(y|g,w)\frac{p(w)p(g)}{q(w)q(g)}. \tag{5.10}$$

We can now use the bound on $p(y|g,w)$ from (5.9) and substitute into (5.10) to obtain

$$p(y) \geq \mathop{\mathbb{E}}_{g,w} \exp\left[\sum_n L_n(g,w_n) - \mathrm{KL}_f\right]\frac{p(w)p(g)}{q(w)q(g)}.$$

Next we use Jensen's inequality for the g expectation. After some rearranging the bound is given by

$$\log p(y) \geq \sum_n \mathbb{E}_g \underbrace{\log \mathbb{E}_w \frac{e^{L_n(g,w_n)}p(w_n)}{q(w_n)}}_{T_n(g)} - \mathrm{KL}_f - \mathrm{KL}_g.$$

To bound $T_n(g)$, we first reduce the variance of the quantity inside the expectation using the sample average as before to tighten the subsequent use of Jensen's inequality. For any K, $T_n(g)$ is (exactly) equal to

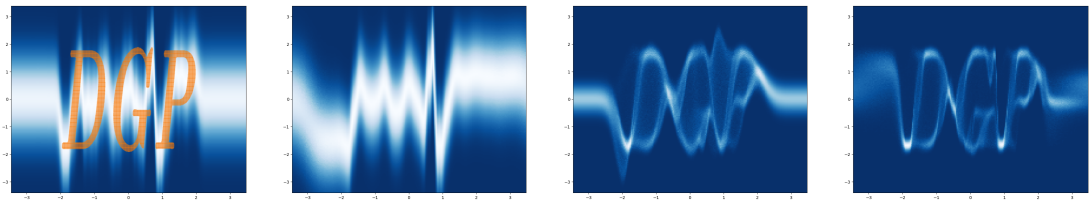
$$T_n(g) = \log \mathbb{E}_{w_n} \frac{1}{K}\sum_k \frac{e^{L_n(g,w_n^{(k)})}p(w_n^{(k)})}{q(w_n^{(k)})},$$

where $w_n^{(k)}$ are independent samples from $q(w_n)$. We now make a final use of Jensen for the w_n expectation, and the final objective is then given by

$$\sum_n \mathbb{E}_{g,w_n} \log \frac{1}{K}\sum_k \frac{e^{L_n(g,w_n^{(k)})}p(w_n^{(k)})}{q(w_n^{(k)})} - \mathrm{KL}_f - \mathrm{KL}_g. \tag{5.11}$$

This bound can be evaluated using Monte Carlo sampling for g and w_n, both with the reparameterization described in Section 5.2.1.

The K terms inside the sum in (5.11) must be sampled with a single draw from $q(g)$, and not independently. This is not an insurmountable problem, however, as we can draw K samples using the full covariance and reparameterize using the Cholesky factor at $\mathcal{O}(K^3)$ cost. Note that the decomposition over the N data points is a consequence of our choice of proposal distributions and the factorization of the likelihood, so we do not incur $\mathcal{O}(N^3)$ complexity and can sample each term in the



(a) GP (b) GP-GP (c) LV-GP (d) LV-GP-GP

Figure 5.4: Posteriors for 1D synthetic data. The models without latent variables cannot capture the bimodality or heteroscedasticity. See Figure 5.1 for the LV-GP-GP-GP model.

sum over N independently.

5.2.4 The posterior over the latent variables

Variational inference simultaneously finds a lower bound and an approximate posterior. The importance weighted approach does also minimize the KL divergence posterior, but the posterior is an implicit one over an augmented space. Samples from this posterior are obtained by first sampling $w_n^k \sim q(w_n)$ and then selecting one of the w_n^k with probabilities proportional to $\frac{e^{L_n(g,w_n^{(k)})}p(w_n^{(k)})}{q(w_n^{(k)})}$. We refer to Domke and Sheldon [2018] for details. In this work, we are not concerned with the posterior over the latent variables themselves, but rather with prediction at test points where we sample from the prior.

5.2.5 Further inference details

The proposal distribution $q(w_n)$ is chosen to be Gaussian $q(w_n) = \mathcal{N}(a_n, b_n)$. We can optimize the ELBO directly with respect to a_n, b_n, or alternatively we can amortize the optimization of these parameters by making them parameterized functions of (x_n, y_n). As the bound is a sum over the data we can use data subsampling. We can also use natural gradients for the variational parameters of the final layer, following Salimbeni et al. [2018b], Dutordoir et al. [2018]. Our bound is modular in both the GP and LV layers so that it extends straightforwardly to the other model variants.

5.3 Results

We use a density estimation task to establish the utility of the DGP models with and without latent variables. For the latent variable models we also compare our importance-weighted (IW) inference (5.11) against the variational inference (VI) approach (5.3). Our central interest is how well the models balance the complexity of the marginals with the complexity of the mapping from inputs to outputs.

5.3.1 1D example

To illustrate the inductive bias of our models we show a 1D example with a conditional density that is non-Gaussian, heteroscedastic and changes discontinuously in some regions of the space. We form a dataset from the letters '*DGP*' by taking a fine grid of points inside the glyphs of a cropped image and using the horizontal coordinates as the input and the vertical coordinates as the output ($73,419$ points in total). Posteriors are shown in Figure 5.4 for four models (the same models as in Figure 5.3), with the data plotted in the first figure. The GP model can obviously not model this data, but neither can the GP-GP as the approximate posterior has no way to separate out the points with the same input but different outputs. The LV-GP model can fit the data to some extent, and the LV-GP-GP model more closely captures the shapes of the letters.

We see that the LV-GP has a tendency to smoother densities than LV-GP-GP, and also a smoother response as a function of the input. Between the '*G*' and '*P*' the LV-GP-GP model extrapolates with high confidence, whereas the LV-GP model maintains a bi-modal distribution connecting the two letters. Posteriors from further models are in Figure 5.4.

5.3.2 UCI datasets

We use 41 publicly available datasets[2] with 1D targets. The datasets range in size from 23 points to $2,049,280$. In each case we reserve 10% of the data for evaluating a test log-likelihood, repeating the experiment five times with different splits. We use five samples for the importance-weighted models, 128 inducing points, and five GP outputs for the inner layers. Hyperparameters and initializations are the same for all models and datasets.Results for test log-likelihood are reported in Table 5.1 for the GP models. Table 5.2 shows additional results for conditional VAE models Sohn et al. [2015b], and results from deep Gaussian process models using stochastic gradient Hamiltonian Monte Carlo Havasi et al. [2018b]. To assess the non-Gaussianity of the predictive distribution we compute the Shapiro Wilk test statistic on the test point marginals. The test statistics are shown in 5.3. Full code to reproduce our results is available online [3]. We draw five conclusions from the results.

1) Latent variables improve performance. For a given depth of Gaussian process mappings (for example GP-GP compared to LV-GP-GP) the latent variable model generally has equal or better performance, and often considerably better. This is true both using IW and VI.

2) IW outperforms VI. The difference is more pronounced on the deeper models, which is to be expected as the deeper models are likely to have more complicated posteriors, so the VI approximation is likely to be more severe. There are also a few datasets ('keggundirected', 'servo' and 'automgp') where the VI latent variable model

[2]The full datasets with the splits and pre-processing can be found at `github.com/hughsalimbeni/bayesian_benchmarks`.

[3]`https://github.com/hughsalimbeni/DGPs_with_IWVI`

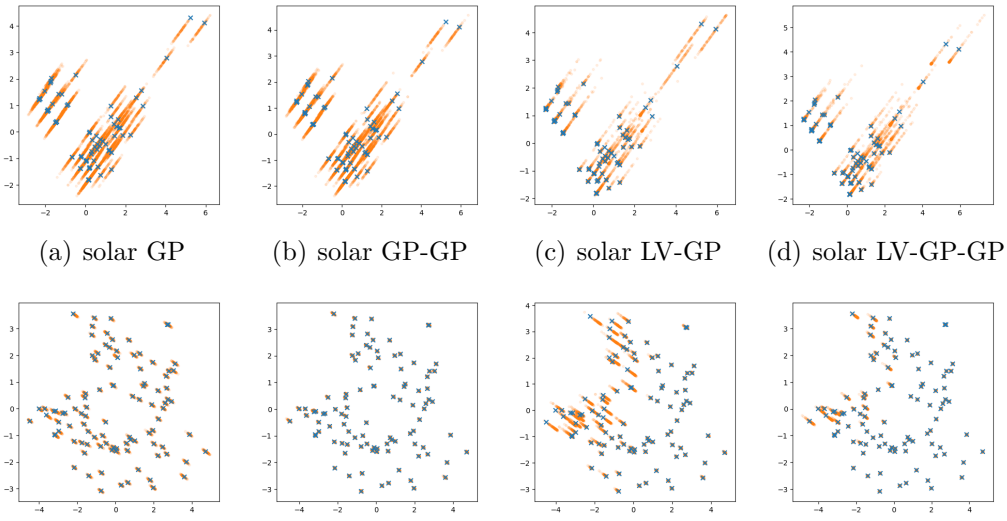
Table 5.1: Average test log-likelihood results for five splits (standard error). **Bold** font indicates overlapping error bars with the best performing method. *Italic* indicates a significantly non-Gaussian posterior, computed by the Shapiro Wilk test (see Table 5.3). Colours indicate datasets demonstrating prominent examples of complex **marginals***, **mappings**†, or **both**‡.

			GP	GP-GP	GP-GP-GP	LV-GP		LV-GP-GP		LV-GP-GP-GP	
Dataset	N	D	VI	VI	VI	VI	IWVI	VI	IWVI	VI	IWVI
challenger	23	4	-3.22 (0.07)	*-2.04 (0.02)*	*-2.10 (0.03)*	***-0.67 (0.01)***	***-0.68 (0.00)***	*-9.99 (3.60)*	*-9.47 (2.19)*	-1.21 (0.01)	*-2.45 (0.04)*
fertility	100	9	-2.51 (0.13)	-1.43 (0.01)	-1.40 (0.01)	-1.41 (0.01)	-1.40 (0.01)	-1.42 (0.01)	-1.69 (0.01)	**-1.32 (0.00)**	**-1.31 (0.01)**
concreteslump	103	7	1.91 (0.01)	1.55 (0.00)	1.45 (0.00)	1.93 (0.00)	**1.94 (0.00)**	1.53 (0.00)	1.75 (0.00)	1.43 (0.00)	1.62 (0.00)
autos	159	25	-0.36 (0.00)	-0.14 (0.01)	-0.14 (0.03)	-0.36 (0.01)	-0.35 (0.00)	**-0.06 (0.04)**	-0.34 (0.01)	-0.25 (0.05)	-0.33 (0.02)
servo	167	4	-0.17 (0.00)	-0.19 (0.01)	-0.17 (0.01)	-0.07 (0.01)	-0.08 (0.01)	-0.19 (0.01)	-0.07 (0.01)	-0.25 (0.01)	**0.03 (0.03)**
breastcancer	194	33	-1.32 (0.00)	-1.36 (0.00)	-1.34 (0.00)	**-1.31 (0.00)**	**-1.31 (0.00)**	-1.43 (0.01)	-1.59 (0.10)	-1.38 (0.00)	-1.80 (0.19)
machine	209	7	-0.70 (0.01)	-0.65 (0.02)	-0.61 (0.01)	-0.75 (0.02)	-0.71 (0.02)	-0.63 (0.01)	**-0.51 (0.01)**	-0.59 (0.01)	**-0.52 (0.03)**
yacht	308	6	1.32 (0.02)	1.84 (0.01)	**2.00 (0.06)**	1.64 (0.00)	1.65 (0.00)	1.69 (0.09)	1.71 (0.01)	1.77 (0.07)	**2.05 (0.02)**
autompg	392	7	-0.44 (0.01)	-0.57 (0.02)	-0.48 (0.03)	-0.32 (0.01)	-0.33 (0.01)	-0.38 (0.04)	**-0.24 (0.01)**	-0.65 (0.11)	-0.28 (0.03)
boston	506	13	0.02 (0.00)	0.07 (0.00)	0.03 (0.01)	-0.04 (0.00)	-0.07 (0.00)	0.06 (0.00)	0.14 (0.00)	-0.04 (0.01)	**0.17 (0.00)**
forest*	517	12	-1.38 (0.00)	-1.37 (0.00)	-1.37 (0.00)	*-0.99 (0.00)*	*-1.00 (0.00)*	***-0.91 (0.00)***	*-1.02 (0.02)*	***-0.92 (0.01)***	***-0.91 (0.01)***
stock	536	11	-0.22 (0.00)	-0.29 (0.00)	-0.26 (0.00)	-0.22 (0.00)	-0.22 (0.00)	-0.29 (0.00)	**-0.19 (0.00)**	-0.23 (0.01)	-0.36 (0.02)
pendulum	630	9	-0.14 (0.00)	0.22 (0.01)	0.12 (0.08)	-0.14 (0.00)	-0.14 (0.00)	0.22 (0.01)	**0.33 (0.08)**	**0.25 (0.00)**	0.20 (0.03)
energy	768	8	1.71 (0.00)	1.85 (0.00)	2.07 (0.01)	1.86 (0.01)	1.92 (0.01)	2.00 (0.04)	1.96 (0.01)	2.07 (0.01)	**2.28 (0.01)**
concrete	1030	8	-0.43 (0.00)	-0.45 (0.00)	-0.35 (0.02)	-0.32 (0.00)	-0.32 (0.00)	-0.35 (0.00)	-0.21 (0.00)	-0.22 (0.01)	**-0.12 (0.00)**
solar*	1066	10	-1.75 (0.08)	-1.21 (0.02)	-1.20 (0.03)	*0.04 (0.07)*	*0.07 (0.01)*	***0.54 (0.01)***	*0.22 (0.01)*	***0.54 (0.01)***	*0.20 (0.02)*
airfoil	1503	5	-0.79 (0.05)	0.08 (0.02)	0.14 (0.03)	-0.44 (0.03)	-0.36 (0.01)	0.07 (0.00)	0.30 (0.00)	-0.02 (0.03)	**0.34 (0.01)**
winered	1599	11	-1.09 (0.00)	-1.11 (0.00)	-1.08 (0.00)	-1.07 (0.00)	-1.06 (0.00)	-1.10 (0.00)	***-0.84 (0.01)***	*-1.06 (0.03)*	*-1.31 (0.40)*
gas	2565	128	1.07 (0.00)	**1.60 (0.05)**	**1.69 (0.05)**	**1.69 (0.08)**	1.56 (0.06)	0.70 (0.10)	**1.57 (0.16)**	1.30 (0.14)	**1.61 (0.12)**
skillcraft	3338	19	-0.94 (0.00)	-0.94 (0.00)	-0.94 (0.00)	**-0.91 (0.00)**	**-0.91 (0.00)**	-0.92 (0.00)	-0.93 (0.00)	-0.92 (0.00)	-0.94 (0.00)
sml†	4137	26	1.53 (0.00)	1.72 (0.01)	1.83 (0.01)	1.52 (0.00)	1.52 (0.00)	1.79 (0.00)	1.91 (0.00)	**1.92 (0.01)**	**1.97 (0.05)**
winewhite	4898	11	-1.14 (0.00)	-1.14 (0.00)	-1.14 (0.00)	-1.13 (0.00)	-1.13 (0.00)	-1.13 (0.00)	**-1.10 (0.00)**	-1.13 (0.00)	**-1.09 (0.00)**
parkinsons	5875	20	1.99 (0.00)	2.61 (0.01)	2.75 (0.02)	1.79 (0.02)	1.82 (0.02)	2.36 (0.02)	2.76 (0.02)	2.71 (0.07)	**3.12 (0.05)**
kin8nm†	8192	8	-0.29 (0.00)	-0.01 (0.00)	-0.00 (0.00)	-0.29 (0.00)	-0.29 (0.00)	-0.02 (0.00)	-0.00 (0.00)	-0.00 (0.00)	**0.03 (0.00)**
pumadyn32nm	8192	32	0.08 (0.00)	**0.11 (0.01)**	**0.11 (0.00)**	-1.44 (0.00)	-1.44 (0.00)	0.10 (0.01)	*-0.62 (0.25)*	**0.11 (0.00)**	**0.11 (0.00)**
power*	9568	4	-0.65 (0.04)	-0.75 (0.03)	-0.80 (0.02)	-0.39 (0.04)	-0.23 (0.02)	-0.36 (0.04)	*-0.28 (0.05)*	-0.25 (0.06)	***-0.11 (0.06)***
naval	11934	14	**4.52 (0.02)**	4.43 (0.03)	4.35 (0.03)	*4.19 (0.01)*	*4.24 (0.01)*	4.36 (0.01)	**4.52 (0.02)**	4.27 (0.02)	4.41 (0.02)
pol‡	15000	26	0.48 (0.00)	1.51 (0.01)	1.45 (0.01)	*0.37 (0.08)*	*-0.50 (0.00)*	*2.34 (0.02)*	***2.63 (0.01)***	1.47 (0.01)	**2.72 (0.10)**
elevators	16599	18	-0.44 (0.00)	-0.41 (0.01)	-0.40 (0.00)	-0.37 (0.00)	-0.36 (0.00)	-0.29 (0.00)	-0.27 (0.00)	-0.28 (0.00)	**-0.27 (0.00)**
bike‡	17379	17	0.82 (0.01)	3.49 (0.01)	3.68 (0.01)	*2.48 (0.01)*	*2.66 (0.01)*	3.48 (0.00)	3.75 (0.01)	3.72 (0.01)	**3.95 (0.01)**
kin40k†	40000	8	0.02 (0.00)	0.84 (0.00)	1.17 (0.00)	0.04 (0.00)	0.05 (0.00)	0.87 (0.01)	0.93 (0.00)	1.15 (0.00)	**1.27 (0.00)**
protein	45730	9	-1.06 (0.00)	-0.98 (0.00)	-0.95 (0.00)	*-0.85 (0.00)*	*-0.80 (0.00)*	*-0.70 (0.00)*	*-0.61 (0.00)*	*-0.68 (0.00)*	***-0.57 (0.00)***
tamielectric	45781	3	-1.43 (0.00)	-1.43 (0.00)	-1.43 (0.00)	***-1.31 (0.00)***	*-1.31 (0.00)*	***-1.31 (0.00)***	*-1.31 (0.00)*	*-1.31 (0.00)*	*-1.31 (0.00)*
keggdirected‡	48827	20	0.16 (0.01)	0.21 (0.01)	0.26 (0.02)	*1.24 (0.06)*	*2.03 (0.01)*	*1.84 (0.05)*	*2.23 (0.01)*	*1.92 (0.02)*	***2.26 (0.01)***
slice†	53500	385	0.86 (0.00)	1.80 (0.00)	1.86 (0.00)	0.85 (0.00)	0.90 (0.00)	1.80 (0.00)	1.88 (0.00)	1.86 (0.00)	**2.02 (0.00)**
keggundirected*	63608	27	0.06 (0.01)	0.06 (0.01)	0.07 (0.01)	*-75 (4)*	*-4.21 (0.33)*	*-64 (6)*	*2.37 (0.29)*	*-116 (17)*	***2.98 (0.21)***
3droad	434874	3	-0.79 (0.00)	-0.61 (0.01)	-0.58 (0.01)	*-0.71 (0.00)*	*-0.65 (0.00)*	*-0.69 (0.00)*	*-0.55 (0.01)*	*-0.61 (0.00)*	***-0.48 (0.00)***
song	515345	90	-1.19 (0.00)	-1.18 (0.00)	-1.18 (0.00)	-1.15 (0.00)	-1.14 (0.00)	-1.13 (0.00)	*-1.10 (0.00)*	-1.12 (0.00)	***-1.07 (0.00)***
buzz	583250	77	-0.24 (0.00)	-0.15 (0.00)	-0.15 (0.00)	-0.22 (0.01)	-0.23 (0.01)	-0.09 (0.01)	-0.04 (0.03)	-0.09 (0.01)	***0.04 (0.00)***
nytaxi	1420068	8	-1.42 (0.01)	-1.44 (0.01)	-1.42 (0.01)	*-1.09 (0.03)*	***-0.90 (0.04)***	*-1.57 (0.04)*	*-1.03 (0.04)*	*-1.74 (0.05)*	***-0.95 (0.07)***
houseelectric‡	2049280	11	1.37 (0.00)	1.41 (0.00)	1.47 (0.01)	*1.65 (0.03)*	*1.82 (0.01)*	1.66 (0.05)	*2.01 (0.00)*	1.77 (0.06)	***2.03 (0.00)***
Median difference from GP baseline			0	0.06	0.09	0.04	0.05	0.12	0.26	0.18	0.32
Average ranks			2.72 (0.14)	4.01 (0.14)	4.84 (0.14)	4.19 (0.17)	4.60 (0.16)	4.84 (0.14)	6.69 (0.15)	5.62 (0.16)	7.49 (0.16)

has very poor performance, but the IW approach performs well. This suggests that the IW approach might be less prone to local optima compared with VI. Recent work [Yao et al., 2018] has examined the quality of variational inference and concluded that it is severely lacking in many practical situations. It is not clear how close either method is to the true posterior, or whether the disparity matters in practice, however.

3) Some datasets require latent variables for any improvement over the single-layer model. For several datasets (for example 'forest' and 'power') the inclusion of latent variables makes a very pronounced difference, whereas depth alone cannot improve over the single layer model. These datasets are highlighted in **blue***. For all these datasets the marginals for the latent variable models are non-Gaussian (see Table 5.3). Conversely, for some datasets (e.g., 'kin40k' and 'sml') we observe much greater benefit from the deep noise free models than the single layer model with latent variables. These datasets are highlighted in **orange**†.

The solar data is a particularly strong example of this. To investigate the posterior in more detail we plotted posterior samples from four models at 100 randomly selected test points, using the 2D PCA projection of the joint space of inputs and outputs in Figure 5.5, top row. We see that both the GP and GP-GP posteriors are the same,



(a) solar GP (b) solar GP-GP (c) solar LV-GP (d) solar LV-GP-GP

(e) bike GP (f) bike GP-GP (g) bike LV-GP (h) bike LV-GP-GP

Figure 5.5: Samples from the predictive distribution at test points (orange) together with the actual value (blue) in the PCA projection to 2D.

but that the latent variable models capture the strongly asymmetric marginals. The LV-GP and LV-GP-GP models differ in the top right part of the projection, where the long tails are missed by the single layer model.

4) Some datasets benefit from both depth and latent variables. For the 'bike' and 'keggdirected' data, for example, we see that both the LV-GP and GP-GP model improve over the single layer model, suggesting that complex input dependence and non-Gaussian marginals are beneficial for modelling the data. Four examples of these datasets are highlighted in **green**[‡]. For these datasets the marginals are non-Gaussian for the latent variable models, and in each case the LV-GP-GP-GP model is the best performing. The bike data shows a striking heavy tailed posterior in some parts of the space. See the lower row of 5.5. We see the that the deeper model provides the skewed posteriors in a smaller region of the space than the LV-GP model, and has tighter posteriors in the bottom right side of the plot. This plot gives a sense of how the LV-GP-GP outperforms both the GP-GP model and LV-GP model.

5) On average, the LV-GP-GP-GP model is best performing. This indicates that the inductive bias of this model is suitable over the broad range of datasets considered. We also compare to Conditional VAE models [Sohn et al., 2015b] (see Table 5.2) and find that these models overfit considerably for the smaller datasets and even on the larger datasets do not outperform our deep latent variable models. The marginals of the CVAE models are more Gaussian than our models (see Table 5.3), indicating that these models explain the data through the input mapping and not the complexity of the marginal densities. See Figures 5.6 and 5.7 for the plots of posterior marginals.

5.4 Related work

If the GP layers are replaced by neural networks our models are equivalent to (conditional) variational autoencoders (VAE) [Kingma and Welling, 2013, Rezende et al., 2014] when used with VI 5.2.1, or importance weighted autoencoders (IWAEs) [Burda et al., 2015] when used with the IW inference 5.2.3. Our model can be thought of as a VAE (or IWAE) but with a deep GP for the 'encoder' mapping. Compared to the conditional VAE, the deep GP approach has several advantages: it incorporates model (epistemic) uncertainty, is automatically regularized, and allows a fine-grained control of the properties of the mapping. For example, the lengthscale corresponding to the latent variable can be tuned to favour complex marginals (a small value) or near Gaussian marginals (a large value). Note that our IW inference is not a simple adaptation of Burda et al. [2015] as we have to perform additional inference over the GPs.

The LV-GP model was proposed by Wang and Neal [2012], and then developed in Dutordoir et al. [2018] and used for meta learning in Sæmundsson et al. [2018]. A version with discrete latent variables was presented in Bodin et al. [2017]. The LV-GP without any inputs is known as the Gaussian process latent variable model [Lawrence, 2004], which was used in a semi-supervised setting in Damianou and Lawrence [2015], which is also equivalent to LV-GP.

The deep model of Damianou and Lawrence [2013a] is closely related to our model but incorporates the latent variables additively rather than through concatenation (that is, $f(g(x_n) + w_n)$ rather than $f(g([x_n, w_n]))$). In principle, it would be possible to recover our model using the approach of Damianou and Lawrence [2013a] in a certain setting of hyperparameters, but in all previous work the kernel hyperparameters were tied within each layer, so this limit was not achievable. **?** also used this model, with a form of expectation propagation for approximate inference.

The variational inference we have presented (without importance weighting) is not equivalent to that of Damianou and Lawrence [2013a]. In Damianou and Lawrence [2013a] a mean-field variational posterior is used for the noisy corruptions, which may be a poor approximation as there are a priori correlations between these outputs. This mean-field assumption also forces independence between the inputs and outputs of the layers, whereas we make no such assumption.

5.5 Discussion

On a broad range of 1D density estimation tasks we find that our DGP with latent variables outperforms the single-layer and noise-free models, sometimes considerably. Closer investigation reveals that non-Gaussian marginals are readily found by our model, and that the importance-weighted objective improves performance in practice.

Conditional density estimation must balance the complexity of the density with the complexity of the input dependency. The inductive bias of our LV-GP-GP-GP

model appears to be suitable for a broad range of datasets we have considered. An advantage of our approach is that the deep models all contain the shallower models as special cases. A layer can be 'turned off' with a single scalar hyperparameter (the kernel variance) set to zero. This is a consequence of the ResNet-inspired [He et al., 2016] use of mean functions. This may explain why we observe empirically that in practice adding depth rarely hurts performance. The latent variables can similarly be 'turned off' if the appropriate lengthscale is large. This may explain why the latent variables models are rarely outperformed by the noise-free models, even when the marginals are Gaussian.

There are very few hyperparameters in our model to optimize (the kernel parameters and the likelihood variance). This allows us to use the same model across a wide range of data. In the small and medium data regimes we have considered, an unregularized mapping (for example, a neural network as in the conditional VAE model [Sohn et al., 2015b]) is likely to overfit, as indeed we have observed.

5.5.1 Limitations

As depth increases it becomes increasingly difficult to reason about the DGP priors. Our approach is unlikely to recover interpretable features, such as periodicity, and the approaches of Lloyd et al. [2014], Sun et al. [2018] may be more appropriate if an interpretable model is required. The latent variables are also difficult to interpret as their effect is coupled with the GP mappings.

We have only considered 1D latent variables and 1D outputs, and while the extension to higher dimensions is straightforward for training, difficulties may arise in evaluating posterior expectations as our model does not provide a closed-form predictive density and Monte Carlo sampling may have unacceptably high variance. The task of estimating test likelihoods with high-dimensional latent variables is challenging, and techniques described in Wu et al. [2016a] may be necessary in higher dimensions.

The inference we have presented is limited by cubic scaling in both K and the number of inducing points. The importance-weighting approach may also suffer from problems of vanishing signal for parameters of $q(w_n)$, as discussed in Rainforth et al. [2018]. The doubly reparameterized gradient estimator from Tucker et al. [2019] could be used to alleviate this problem.

5.6 Conclusion

We have presented a novel inference scheme for the deep Gaussian process with latent variables, combining importance weighting with partially collapsed variational inference. We have also developed a variant of the deep Gaussian process model where uncorrelated variables are introduced as latent covariates rather than process noise. We have shown empirically that latent variables models deep models outperform the noise-free deep GP on a range of data, and also that our importance-weighted inference delivers an advantage over variational inference in practice.

Dataset	N	D	Linear VI	CVAE 50 VI	CVAE 100, 100 VI	GP VI	GP-GP VI	GP-GP-GP VI	GP SGHMC	GP-GP SGHMC	GP-GP-GP SGHMC	LV-GP VI	LV-GP IWVI	LV-GP-GP VI	LV-GP-GP IWVI	LV-GP-GP-GP VI	LV-GP-GP-GP IWVI
challenger	23	4	-1.12 (0.02)	***-0.59 (0.03)***	$-\infty$	-3.22 (0.07)	*-2.04 (0.02)*	*-2.10 (0.03)*	$-\infty$	$-\infty$	*-654 (337)*	*-0.67 (0.01)*	*-0.68 (0.00)*	*-9.99 (3.60)*	*-9.47 (2.19)*	-1.21 (0.01)	*-2.45 (0.04)*
fertility	100	9	**-1.32 (0.01)**	$-\infty$	$-\infty$	-2.51 (0.13)	-1.43 (0.01)	-1.40 (0.01)	-2.19 (0.02)	*-1.67 (0.03)*	*-1.35 (0.02)*	-1.41 (0.01)	-1.40 (0.01)	-1.42 (0.01)	-1.69 (0.01)	**-1.32 (0.00)**	**-1.31 (0.01)**
concreteslump	103	7	0.71 (0.01)	$-\infty$	$-\infty$	1.91 (0.01)	1.55 (0.00)	1.45 (0.00)	0.59 (0.04)	0.10 (0.03)	-0.14 (0.03)	1.93 (0.00)	**1.94 (0.00)**	1.53 (0.00)	1.75 (0.00)	1.43 (0.00)	1.62 (0.00)
autos	159	25	-0.43 (0.00)	$-\infty$	$-\infty$	-0.36 (0.00)	-0.14 (0.01)	-0.14 (0.03)	-10 (1)	-15 (2)	*-23 (4)*	-0.36 (0.01)	-0.35 (0.00)	**-0.06 (0.04)**	-0.34 (0.01)	-0.25 (0.05)	-0.33 (0.02)
servo	167	4	-0.86 (0.00)	*-385 (220)*	$-\infty$	-0.17 (0.00)	-0.19 (0.01)	-0.17 (0.01)	-1.22 (0.21)	-27 (10)	-326 (44)	-0.07 (0.01)	-0.08 (0.01)	-0.19 (0.01)	-0.07 (0.01)	-0.25 (0.01)	**0.03 (0.03)**
breastcancer	194	33	-1.49 (0.01)	$-\infty$	$-\infty$	-1.32 (0.00)	-1.36 (0.00)	-1.34 (0.00)	-10 (1)	*-28 (10)*	*-462 (100)*	**-1.31 (0.00)**	**-1.31 (0.00)**	-1.43 (0.01)	-1.59 (0.10)	-1.38 (0.00)	-1.80 (0.19)
machine	209	7	-0.71 (0.02)	*-114 (78)*	$-\infty$	-0.70 (0.01)	-0.65 (0.02)	-0.61 (0.01)	-2.21 (0.28)	-14 (4)	-74 (13)	-0.75 (0.02)	-0.71 (0.02)	-0.63 (0.01)	**-0.51 (0.01)**	-0.59 (0.01)	**-0.52 (0.03)**
yacht	308	6	-0.14 (0.10)	-7.91 (4.34)	$-\infty$	1.32 (0.02)	1.84 (0.01)	**2.00 (0.06)**	-0.26 (0.79)	-9.70 (1.35)	-16 (2)	1.64 (0.00)	1.65 (0.00)	1.69 (0.09)	1.71 (0.01)	1.77 (0.07)	**2.05 (0.02)**
autompg	392	7	-0.48 (0.00)	*-71 (9)*	$-\infty$	-0.44 (0.01)	-0.57 (0.02)	-0.48 (0.03)	-2.35 (0.22)	-43 (4)	-131 (26)	-0.32 (0.01)	-0.33 (0.01)	-0.38 (0.04)	**-0.24 (0.01)**	-0.65 (0.11)	-0.28 (0.03)
boston	506	13	-0.56 (0.00)	$-\infty$	$-\infty$	0.02 (0.00)	0.07 (0.00)	0.03 (0.01)	0.14 (0.00)	-2.70 (0.70)	-21 (5)	-0.04 (0.00)	-0.07 (0.00)	0.06 (0.00)	0.14 (0.00)	-0.04 (0.01)	**0.17 (0.00)**
forest*	517	12	-1.36 (0.00)	*-47 (15)*	$-\infty$	-1.38 (0.00)	-1.37 (0.00)	-1.37 (0.00)	-1.75 (0.02)	-3.24 (0.48)	-43 (5)	*-0.99 (0.00)*	*-1.00 (0.00)*	***-0.91 (0.00)***	*-1.02 (0.02)*	***-0.92 (0.01)***	***-0.91 (0.01)***
stock	536	11	**-0.16 (0.00)**	$-\infty$	$-\infty$	-0.22 (0.00)	-0.29 (0.00)	-0.26 (0.00)	-0.22 (0.01)	-2.46 (0.41)	-97 (13)	-0.22 (0.00)	-0.22 (0.00)	-0.29 (0.00)	-0.19 (0.00)	-0.23 (0.01)	-0.36 (0.02)
pendulum	630	9	-1.16 (0.00)	$-\infty$	$-\infty$	-0.14 (0.00)	0.22 (0.01)	0.12 (0.08)	-0.11 (0.01)	**0.33 (0.06)**	-5.43 (1.52)	-0.14 (0.00)	-0.14 (0.00)	0.22 (0.01)	**0.33 (0.08)**	**0.25 (0.00)**	0.20 (0.03)
energy	768	8	-0.06 (0.00)	-0.44 (0.31)	$-\infty$	1.71 (0.00)	1.85 (0.00)	2.07 (0.01)	1.75 (0.00)	1.97 (0.01)	1.12 (0.33)	1.86 (0.01)	1.92 (0.01)	2.00 (0.04)	1.96 (0.01)	2.07 (0.01)	**2.28 (0.01)**
concrete	1030	8	-0.97 (0.00)	-26 (10)	$-\infty$	-0.43 (0.00)	-0.45 (0.00)	-0.35 (0.02)	-0.33 (0.01)	-0.88 (0.38)	-1.85 (1.01)	-0.32 (0.00)	-0.32 (0.00)	-0.35 (0.00)	-0.21 (0.00)	-0.22 (0.01)	**-0.12 (0.00)**
solar*	1066	10	-1.54 (0.03)	***0.71 (0.02)***	*0.26 (0.34)*	-1.75 (0.08)	-1.21 (0.02)	-1.20 (0.03)	-1.56 (0.05)	-1.58 (0.08)	-1.65 (0.06)	*0.04 (0.07)*	*0.07 (0.01)*	*0.54 (0.01)*	*0.22 (0.01)*	*0.54 (0.01)*	*0.20 (0.02)*
airfoil	1503	5	-1.11 (0.00)	-0.64 (0.32)	$-\infty$	-0.79 (0.05)	0.08 (0.02)	0.14 (0.03)	-0.90 (0.05)	-0.19 (0.20)	**0.32 (0.09)**	-0.44 (0.03)	-0.36 (0.01)	0.07 (0.00)	0.30 (0.00)	-0.02 (0.03)	**0.34 (0.01)**
winered	1599	11	-1.14 (0.00)	*-9.64 (1.40)*	*-1.86 (0.85)*	-1.09 (0.00)	-1.11 (0.00)	-1.08 (0.00)	-1.11 (0.00)	-1.09 (0.01)	-1.63 (0.09)	-1.07 (0.00)	-1.06 (0.00)	-1.10 (0.00)	***-0.84 (0.01)***	*-1.06 (0.03)*	*-1.31 (0.40)*
gas	2565	128	0.23 (0.00)	-241 (77)	$-\infty$	1.07 (0.00)	**1.60 (0.05)**	**1.69 (0.05)**	0.93 (0.01)	1.20 (0.02)	1.46 (0.05)	**1.69 (0.08)**	1.56 (0.06)	0.70 (0.10)	**1.57 (0.16)**	1.30 (0.14)	**1.61 (0.12)**
skillcraft	3338	19	-0.95 (0.00)	-5.80 (0.29)	$-\infty$	-0.94 (0.00)	-0.94 (0.00)	-0.94 (0.00)	-0.95 (0.00)	-0.99 (0.02)	-1.10 (0.03)	**-0.91 (0.00)**	**-0.91 (0.00)**	-0.92 (0.00)	-0.93 (0.00)	-0.92 (0.00)	-0.94 (0.00)
sml†	4137	26	0.32 (0.00)	-3.79 (0.87)	$-\infty$	1.53 (0.00)	1.72 (0.01)	1.83 (0.01)	1.46 (0.00)	1.71 (0.04)	**1.93 (0.01)**	1.52 (0.00)	1.52 (0.00)	1.79 (0.00)	1.91 (0.00)	**1.92 (0.01)**	**1.97 (0.05)**
winewhite	4898	11	-1.22 (0.00)	-1.37 (0.04)	$-\infty$	-1.14 (0.00)	-1.14 (0.00)	-1.14 (0.00)	-1.13 (0.00)	-1.13 (0.00)	-1.14 (0.01)	-1.13 (0.00)	-1.13 (0.00)	-1.13 (0.00)	**-1.10 (0.00)**	-1.13 (0.00)	**-1.09 (0.00)**
parkinsons	5875	20	-1.30 (0.00)	-1.14 (0.56)	$-\infty$	1.99 (0.00)	2.61 (0.01)	2.75 (0.02)	1.87 (0.02)	2.38 (0.05)	2.41 (0.04)	1.79 (0.02)	1.82 (0.02)	2.36 (0.02)	2.76 (0.02)	2.71 (0.07)	**3.12 (0.05)**
kin8nm†	8192	8	-1.12 (0.00)	-0.01 (0.01)	$-\infty$	-0.29 (0.00)	-0.01 (0.00)	-0.00 (0.00)	-0.30 (0.00)	-0.03 (0.01)	-0.00 (0.00)	-0.29 (0.00)	-0.29 (0.00)	-0.02 (0.00)	-0.00 (0.00)	-0.00 (0.00)	**0.03 (0.00)**
pumadyn32nm	8192	32	-1.44 (0.00)	-2.10 (0.09)	$-\infty$	0.08 (0.00)	0.11 (0.01)	0.11 (0.00)	**0.11 (0.00)**	**0.11 (0.00)**	0.10 (0.00)	-1.44 (0.00)	-1.44 (0.00)	0.10 (0.01)	*-0.62 (0.25)*	0.11 (0.00)	0.11 (0.00)
power*	9568	4	-0.43 (0.01)	-0.44 (0.08)	-2.14 (0.06)	-0.65 (0.04)	-0.75 (0.03)	-0.80 (0.02)	-0.58 (0.03)	-1.41 (0.00)	-1.41 (0.00)	-0.39 (0.04)	-0.23 (0.02)	-0.36 (0.04)	*-0.28 (0.05)*	-0.25 (0.06)	***-0.11 (0.06)***
naval	11934	14	-0.63 (0.00)	3.07 (0.08)	3.39 (0.12)	**4.52 (0.02)**	4.43 (0.03)	4.35 (0.03)	3.17 (0.03)	-1.41 (0.00)	-1.41 (0.00)	*4.19 (0.01)*	*4.24 (0.01)*	4.36 (0.01)	**4.52 (0.02)**	4.27 (0.02)	4.41 (0.02)
pol‡	15000	26	-1.10 (0.00)	-0.05 (0.08)	-411 (19)	0.48 (0.00)	1.51 (0.01)	1.45 (0.01)	0.46 (0.00)	1.32 (0.03)	1.31 (0.07)	*0.37 (0.08)*	*-0.50 (0.00)*	*2.34 (0.02)*	***2.63 (0.01)***	1.47 (0.01)	**2.72 (0.10)**
elevators	16599	18	-0.74 (0.00)	-0.28 (0.00)	-33 (1)	-0.44 (0.00)	-0.41 (0.01)	-0.40 (0.00)	-0.46 (0.00)	-0.40 (0.01)	-0.39 (0.01)	-0.37 (0.00)	-0.36 (0.00)	-0.29 (0.00)	-0.27 (0.00)	-0.28 (0.00)	**-0.27 (0.00)**
bike‡	17379	17	-0.76 (0.00)	*3.90 (0.01)*	**4.47 (0.02)**	0.82 (0.01)	3.49 (0.01)	3.68 (0.01)	1.06 (0.01)	1.21 (0.95)	2.77 (0.10)	*2.48 (0.01)*	*2.66 (0.01)*	3.48 (0.00)	3.75 (0.01)	3.72 (0.01)	3.95 (0.01)
kin40k†	40000	8	-1.43 (0.00)	0.65 (0.01)	**1.24 (0.13)**	0.02 (0.00)	0.84 (0.00)	1.17 (0.00)	-0.02 (0.00)	-1.43 (0.00)	-1.43 (0.00)	0.04 (0.00)	0.05 (0.00)	0.87 (0.01)	0.93 (0.00)	1.15 (0.00)	**1.27 (0.00)**
protein	45730	9	-1.25 (0.00)	*-0.78 (0.02)*	*-0.57 (0.00)*	-1.06 (0.00)	-0.98 (0.00)	-0.95 (0.00)	-1.08 (0.00)	-1.10 (0.07)	-0.95 (0.00)	*-0.85 (0.00)*	*-0.80 (0.00)*	*-0.70 (0.00)*	*-0.61 (0.00)*	*-0.68 (0.00)*	***-0.57 (0.00)***
tamielectric	45781	3	-1.43 (0.00)	*-1.31 (0.00)*	***-1.31 (0.00)***	-1.43 (0.00)	-1.43 (0.00)	-1.43 (0.00)	-1.43 (0.00)	-1.43 (0.00)	-1.43 (0.00)	*-1.31 (0.00)*	*-1.31 (0.00)*	*-1.31 (0.00)*	*-1.31 (0.00)*	*-1.31 (0.00)*	*-1.31 (0.00)*
keggdirected‡	48827	20	-0.34 (0.15)	*0.65 (0.32)*	*1.59 (0.14)*	0.16 (0.01)	0.21 (0.01)	0.26 (0.02)	0.26 (0.02)	0.45 (0.02)	0.53 (0.04)	*1.24 (0.06)*	*2.03 (0.01)*	*1.84 (0.05)*	*2.23 (0.01)*	*1.92 (0.02)*	***2.26 (0.01)***
slice†	53500	385	-0.47 (0.00)	-27 (1)	-358 (15)	0.86 (0.00)	1.80 (0.00)	1.86 (0.00)	1.02 (0.00)	1.84 (0.02)	1.93 (0.01)	0.85 (0.00)	0.90 (0.00)	1.80 (0.00)	1.88 (0.00)	1.86 (0.00)	**2.02 (0.00)**
keggundirected*	63608	27	0.28 (0.00)	*-344 (39)*	$-\infty$	0.06 (0.01)	0.06 (0.01)	0.07 (0.01)	-0.03 (0.02)	-0.14 (0.23)	0.10 (0.01)	*-75 (4)*	*-4.21 (0.33)*	*-64 (6)*	*2.37 (0.29)*	*-116 (17)*	***2.98 (0.21)***
3droad	434874	3	-1.41 (0.00)	*-0.85 (0.00)*	*-0.53 (0.01)*	-0.79 (0.00)	-0.61 (0.01)	-0.58 (0.01)	-1.15 (0.06)	-1.42 (0.00)	-1.42 (0.00)	*-0.71 (0.00)*	*-0.65 (0.00)*	*-0.69 (0.00)*	*-0.55 (0.01)*	*-0.61 (0.00)*	***-0.48 (0.00)***
song	515345	90	-1.28 (0.00)	-1.11 (0.00)	-1.09 (0.00)	-1.19 (0.00)	-1.18 (0.00)	-1.18 (0.00)	-1.19 (0.00)	-1.17 (0.00)	-1.17 (0.00)	-1.15 (0.00)	-1.14 (0.00)	-1.13 (0.00)	*-1.10 (0.00)*	-1.12 (0.00)	***-1.07 (0.00)***
buzz	583250	77	-1.56 (0.04)	0.01 (0.00)	**0.07 (0.00)**	-0.24 (0.00)	-0.15 (0.00)	-0.15 (0.00)	-0.92 (0.02)	-0.48 (0.01)	-0.43 (0.03)	-0.22 (0.01)	-0.23 (0.01)	-0.09 (0.01)	-0.04 (0.03)	-0.09 (0.01)	*0.04 (0.00)*
nytaxi	1420068	8	-1.53 (0.01)	*-1.64 (0.04)*	*-1.90 (0.04)*	-1.42 (0.01)	-1.44 (0.01)	-1.42 (0.01)	-1.46 (0.01)	-1.78 (0.00)	-1.77 (0.00)	*-1.09 (0.03)*	***-0.90 (0.04)***	*-1.57 (0.04)*	*-1.03 (0.04)*	*-1.74 (0.05)*	***-0.95 (0.07)***
houseelectric‡	2049280	11	-0.52 (0.00)	1.41 (0.00)	1.48 (0.01)	1.37 (0.00)	1.41 (0.00)	1.47 (0.01)	0.01 (0.04)	-1.43 (0.00)	-1.43 (0.00)	*1.65 (0.03)*	*1.82 (0.01)*	1.66 (0.05)	*2.01 (0.00)*	1.77 (0.06)	***2.03 (0.00)***
Median difference from GP baseline			-0.38	-2.06	-6016.67	0	0.06	0.09	-0.02	-0.02	-0.28	0.04	0.05	0.12	0.26	0.18	0.32
Average ranks			4.99 (0.25)	5.32 (0.31)	4.35 (0.38)	6.80 (0.21)	8.65 (0.20)	9.63 (0.20)	5.57 (0.19)	5.57 (0.24)	5.21 (0.28)	8.94 (0.22)	9.36 (0.22)	9.82 (0.17)	12.04 (0.18)	10.71 (0.19)	13.03 (0.20)

Table 5.2: Test log likelihood values over 5 splits (standard errors), including the CVAE models and models with Stochastic Gradient HMC Havasi et al. [2018b]. Results less than -1000 are reported as $-\infty$. The numbers after the CVAE models are the number of hidden units. The SGHMC results were run with identical models as the corresponding VI methods, Hyperparameter optimization was performed using a random sample from the last 100 iterations, as described in Havasi et al. [2018b]. Posterior sample were take with 2000 samples, with a thinning factor of 5 (i.e. a chain of length 10000). We included also a single layer GP using the SGHMC approach for comparison. The discrepancy between the variational approach and the SGHMC is attributable to several factors (note that the variational approach is optimal in this case): different batch sizes (512 in this work compared to 10000 in Havasi et al. [2018b]); different hyperparameter initializations (for example, a much longer initial lengthscales is used in Havasi et al. [2018b]); different learning rate schedules (decaying learning rate in this work compared to fixed in Havasi et al. [2018b]); the use of natural gradients in this work for the final layer, as opposed to optimizing with Adam optimizer, as done in Havasi et al. [2018b], Salimbeni and Deisenroth [2017b]; insufficient mixing in the posterior chain.

Dataset	N	D	Linear VI	CVAE 50 VI	CVAE 100, 100 VI	GP VI	GP-GP VI	GP-GP-GP VI	GP SGHMC	GP-GP SGHMC	GP-GP-GP SGHMC	LV-GP VI	LV-GP IWVI	LV-GP-GP VI	LV-GP-GP IWVI	LV-GP-GP-GP VI	LV-GP-GP-GP IWVI
challenger	23	4		0.5191		0.9990	0.8870	0.7739		0.9823	0.9099	0.7001	0.7115	0.8466	0.8484		0.7187
fertility	100	9		0.9408	0.8353					0.9598	0.9341				0.9928		0.9969
concreteslump	103	7								0.9970	0.9922						
autos	159	25		0.9823	0.8798					0.9903	0.9874				0.9990		0.9990
servo	167	4		0.8358						0.9989	0.9975	0.9990	0.9990				0.9988
breastcancer	194	33								0.9787	0.9814				0.9911		0.9943
machine	209	7		0.9181	0.8967					0.9974	0.9989						
yacht	308	6		0.9987											0.9989		0.9990
autompg	392	7		0.9097	0.9948					0.9989	0.9987	0.9990	0.9988	0.9987	0.9985		0.9988
boston	506	13		0.9916							0.9989	0.9989	0.9979	0.9989	0.9980		0.9981
forest*	517	12		0.8453	0.7145					0.9981	0.9964	0.7034	0.7074	0.7470	0.7398	0.7608	0.7642
stock	536	11															
pendulum	630	9													0.9989		0.9988
energy	768	8											0.9987				0.9990
concrete	1030	8		0.9986	0.9782										0.9990		0.9989
solar*	1066	10		0.3028	0.3228							0.3380	0.3509	0.2728	0.2945	0.2557	0.2883
airfoil	1503	5		0.9989	0.9986										0.9988		0.9988
winered	1599	11		0.9790	0.6980										0.9234	0.9751	0.8473
gas	2565	128		0.9984								0.9983	0.9935	0.9989	0.9983	0.9990	0.9986
skillcraft	3338	19										0.9989	0.9983	0.9989	0.9988	0.9990	0.9989
sml†	4137	26															
winewhite	4898	11													0.9990		0.9990
parkinsons	5875	20		0.9989													
kin8nm†	8192	8															
pumadyn32nm	8192	32													0.9770		
power*	9568	4		0.9964								0.9963	0.9926	0.9964	0.9852	0.9963	0.9855
naval	11934	14										0.9649	0.9332				
pol‡	15000	26		0.9970								0.8482	0.7214	0.9839	0.9508		0.9980
elevators	16599	18		0.9979								0.9986	0.9977	0.9987	0.9976	0.9988	0.9979
bike‡	17379	17		0.9815								0.9657	0.9337	0.9990	0.9990		
kin40k†	40000	8		0.9990													
protein	45730	9		0.8839	0.8800							0.9117	0.9041	0.8904	0.8761	0.8840	0.8691
tamielectric	45781	3		0.9551	0.9502							0.9558	0.9562	0.9550	0.9549	0.9551	0.9549
keggdirected‡	48827	20		0.8279	0.5350							0.7498	0.5445	0.6839	0.5187	0.7039	0.5001
slice†	53500	385										0.9990	0.9983		0.9988		0.9988
keggundirected*	63608	27		0.4804	0.9882							0.5287	0.3281	0.5511	0.2049	0.9295	0.2322
3droad	434874	3		0.9421	0.9479							0.9432	0.9271	0.9650	0.9467	0.9744	0.9464
song	515345	90		0.9936	0.9957							0.9983	0.9932	0.9969	0.9893	0.9963	0.9835
buzz	583250	77		0.9952	0.9909									0.9988	0.9938	0.9984	0.9838
nytaxi	1420068	8		0.9608	0.9422							0.9698	0.9274	0.9515	0.9019	0.9467	0.8981
houseelectric‡	2049280	11										0.9830	0.9676	0.9962	0.9671	0.9933	0.9743

Table 5.3: Median Shapiro Wilk test statistic for the test points. Blanks have values of one (perfectly Gaussian). Smaller values indicate more evidence for non-Gaussian marginal distributions.

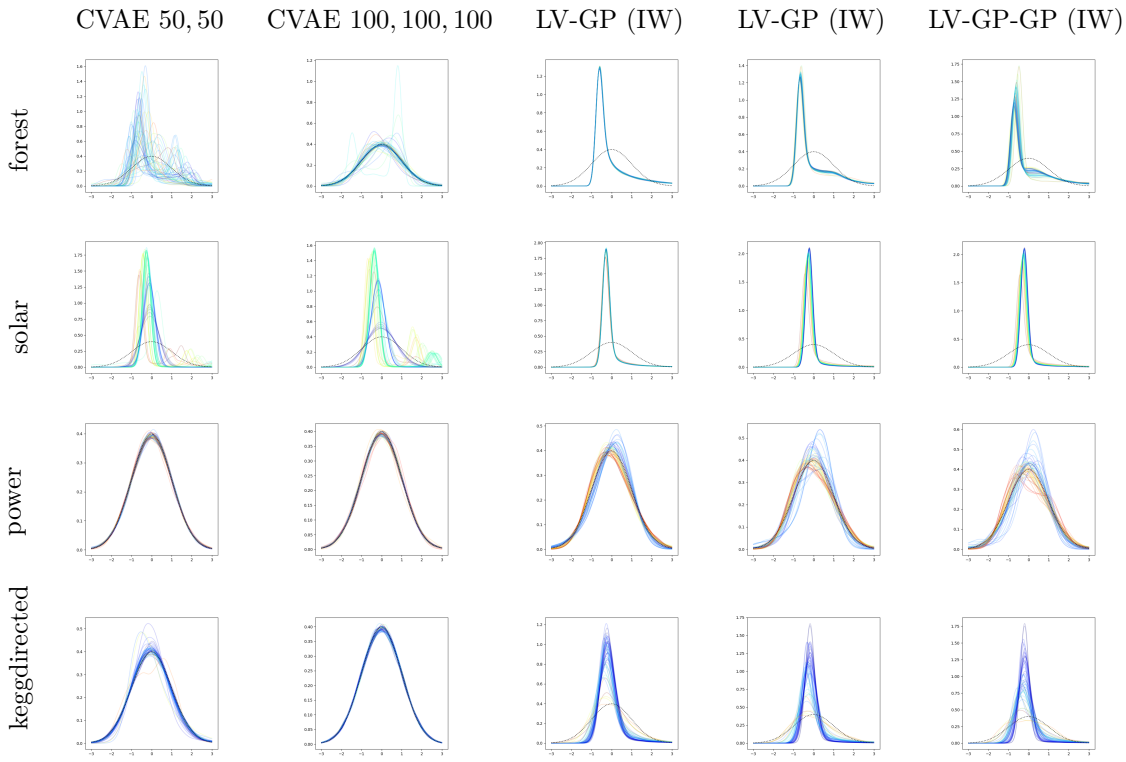


Figure 5.6: Posterior marginals, re-scaled to zero mean and unit standard deviation, with the Gaussian density marked as a dotted curve. Colors are according to the principal component of the inputs. These are the four datasets highlighted in the main text for being prominent examples benefiting from latent variables. In these cases the additional depth of model leads to more non-Gaussian marginals.

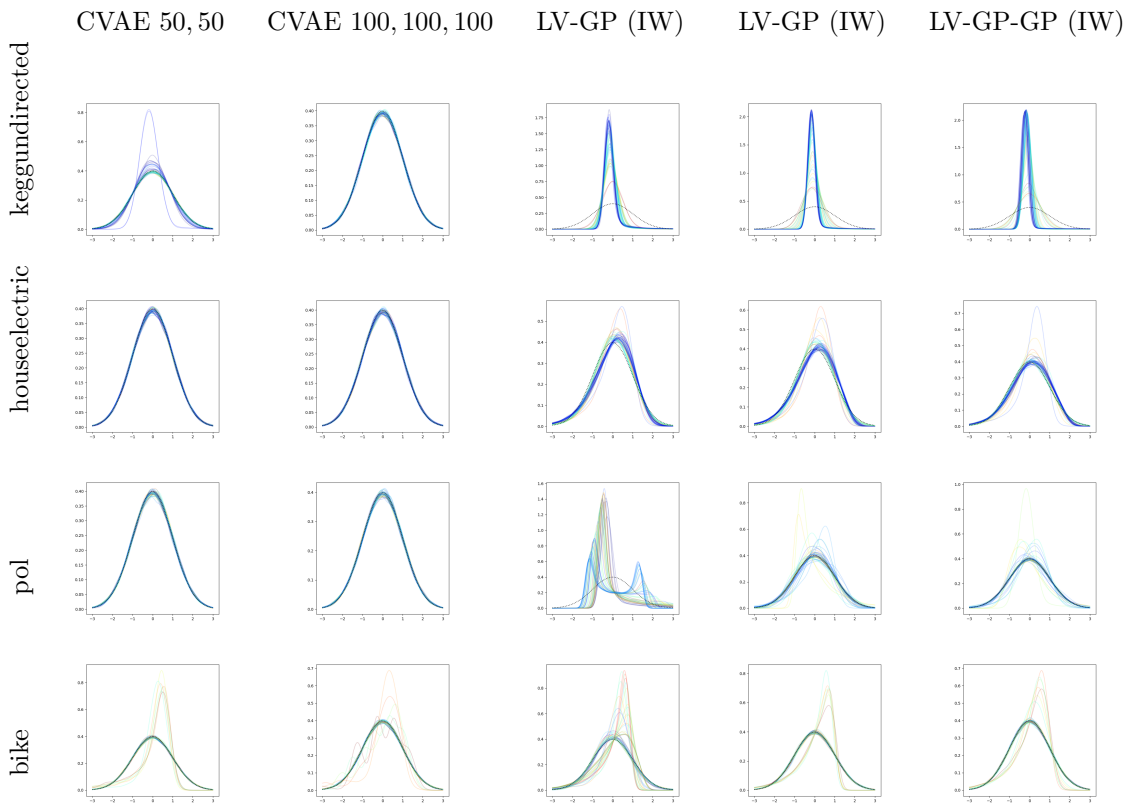


Figure 5.7: As Figure 5.6, but for the four datasets highlighted in the main text for benefiting from both latent variables and depth. Note that, unlike in Figure 5.6, for the 'pol' and 'bike' data the deeper models actually have *simpler* marginals. This is because the deep model has more complexity in the mapping, so can more closely fit the data, whereas the simple model must explain the data with a complex noise distribution.

Chapter 6

Impact

The impact of the work presented in thesis is demonstrated by subsequent work from other groups developing and applying the ideas. Since the original publication of the ideas of Chapter 2 in Salimbeni and Deisenroth [2017b], the approach has been extended and developed in various ways. This section presents some of developments extant as of June 2019 (with any luck this chapter will become out of date...).

Inference Extensions The doubly-stochastic variational approach for inference has been extended in at least two directions. In Havasi et al. [2018a], the inducing point distributions (that is, $q(\tilde{\mathbf{f}}^{\mathbf{1}})q(\tilde{\mathbf{f}}^{\mathbf{2}})$ for a two layer model) are not assumed Gaussian but are directly sampled over using stochastic gradient Hamiltonian Monte Carlo (HMC). The HMC approach uses a free-form $q(\tilde{\mathbf{f}}^1, \tilde{\mathbf{f}}^2)$ rather than a Gaussian. In this case it is possible to express the optimal log density for $q(\tilde{\mathbf{f}}^1, \tilde{\mathbf{f}}^2)$ in terms of an expectation over $q(f^1|\tilde{\mathbf{f}}^1)q(f^2|\tilde{\mathbf{f}}^2)$. This method in the single layer case was presented in **?**. The calculation of this density in the deep model is essentially identical to that presented in Chapter 2, but the variational posteriors take the form of (1.17) and (1.18). The advantage of this approach is that the posteriors are coupled between layers. Results in Havasi et al. [2018a] suggest that the method works well in practice.

A new generalized approach to variational inference was presented in Knoblauch et al. [2019]. In this work both the log-likelihood expectations and the KL terms are modified to make them more 'robust' by penalizing outliers less strongly than in the Gaussian case. The idea is applied to the deep GP model of Chapter 2 as a case study, and results are presented that demonstrate the superiority of the method.

Convolutional Layers An appealing avenue of further research is to incorporate convolutional layers suitable for image data, and this has been pursued by a number of groups (see for example Dutordoir et al. [2019], Kumar et al. [2018], Blomqvist et al. [2018]). There are computational difficulties, however. The number of outputs at each layer needs to be quite large, and the outputs cannot be taken as independent as the convolutional structure shares the same function over multiple outputs. The sampling approach presented in chapter 2 is still valid in the convolutional case, but the independence is over the N data, not the D outputs at each layer. For coupled outputs the sampling would have to scale like $\mathcal{O}(D^3)$, which for a 100×100 image with stride of 1 leads and padding leads to $D = 10,000$, which is prohibitively large. It is possible to ignore these correlations or make further simplifying assumptions as in Kumar et al. [2018], Blomqvist et al. [2018]. These methods appears to work well in practice.

Multi-fidelity modelling Multi-fidelity methods deal with the setting where data comes in two categories: cheaply obtained but biased and noisy data, and expensive noise-free data. The aim is to combine the expensive data with the cheap data to construct a reliable model. An example application might be spatio−temporal monitoring of CO_2 levels in the atmosphere: reliable measurements with calibrated equipment can only be obtained at restricted locations, but approximate measurements using low quality sensors are more plentifully obtained. A deep Gaussian process can be used in this setting to account for the non-linear misalignment of the low-fidelity observations with the high-fidelity observations. Cutajar et al. [2019] used the DGP in this way, and used inference similar to that presented in Chapter 2.

Bayesian Optimization The first use of a DGP in the context of Bayesian optimization is presented in Dutordoir [2017], using the methods of Damianou and Lawrence [2013a]. Using the work presented in Chapter 2, Hebbal et al. [2019] used the DGP as a surrogate function in a range of tasks, including the optimization of a parameters for a rocket booster simulator. See Figure 6.1 for the rocket parameters, and Figure 6.2 for sectional views to demonstrate the non-stationary nature of the problem. Results in Hebbal et al. [2019] show that the DGP offers advantages in this regime relative to the single layer GP model.

Applications There have been a number of applications of the approach of Chapter 2 since the original publication in May 2017. For example, Koriyama and Kobayashi [2019] used the method to synthesize audio Japanese phrases. Samples can be heard at `http://www.kbys.ip.titech.ac.jp/demo/dgpss/`. The DGP model is found to compare favourable to deep neural networks. See Figure 6.3 for a quantitative comparison. The method additionally used ArcCosine kernels (introduced in 1.1).

In another application, Svendsen et al. [2018] used DGPs for geophysical parameter retrieval. The data used is spectrogram data from Aires et al. [2002], and the results show the DGP improving with depth. See Figure 6.4(a) for the results of RMSE with varying number of training data, and 6.4(b) for an plot of illustrative spectrogram

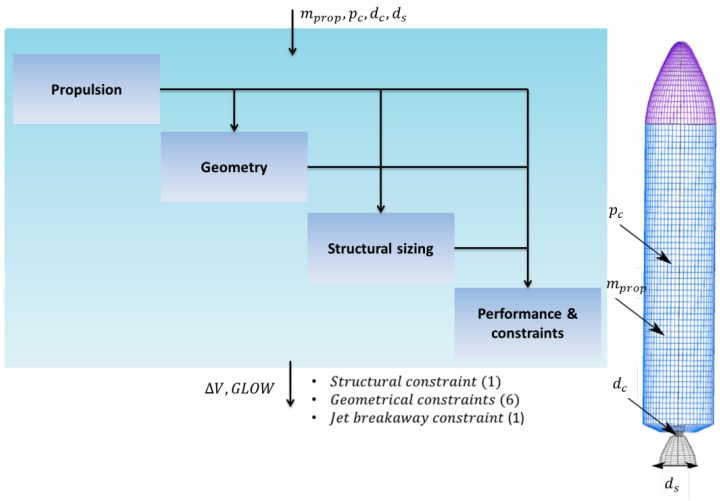


Figure 6.1: Figure from Hebbal et al. [2019], showing the variables to be optimized using the DGP as a surrogate function within the Bayesian Optimization framework

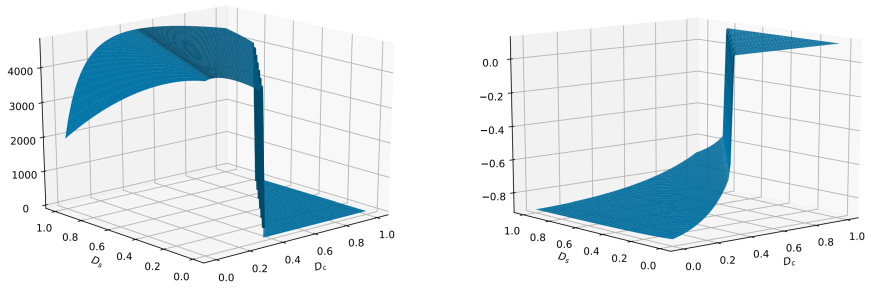


Figure 6.2: Figures from Hebbal et al. [2019], showing a sectional view of the change in velocity according to the diameters of the nozzle of the rocket booster in Figure 6.1, demonstrating the non-stationary behaviours of the functions involved in the optimization problem.

data (taken from Aires et al. [2002]). The gap between the deeper models widens as there is more data, illustrating the advantages of the more flexible deep model. Note that, similarly to the observations made in Chapters 2 and 5, the deep models do not overfit with small data, and recover the single layer model.

The doubly stochastic approach of Chapter 2 was also used for contaminant source localization in Park et al. [2018] in combination with an Deep Boltzmann Machine (DBM).

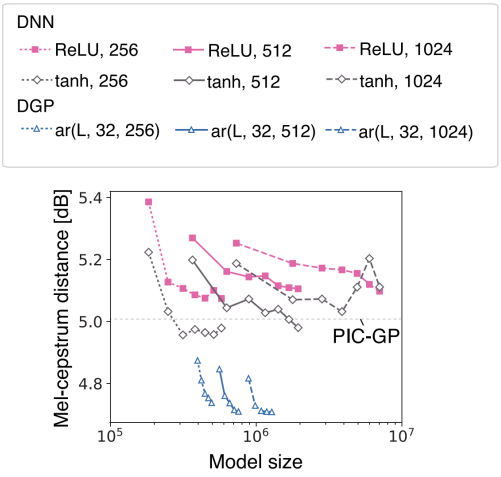
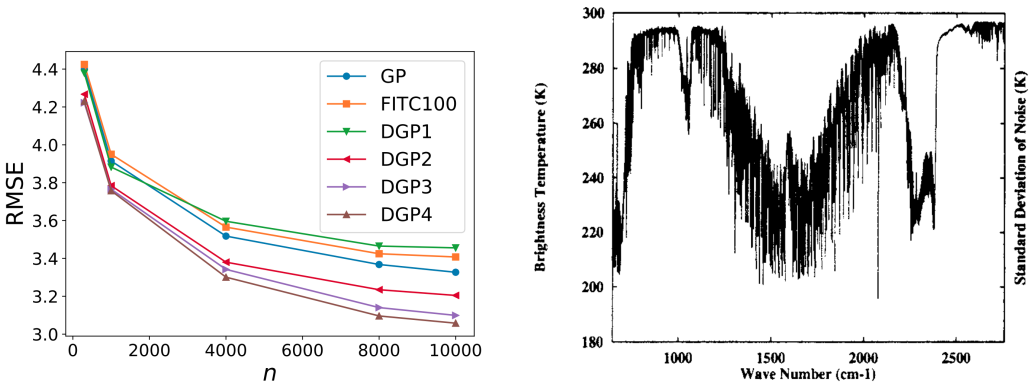


Figure 6.3: Figure from Koriyama and Kobayashi [2019], showing the quantitative improvements of the DGP model in speech synthesis. The mel-cepstrum distance is a metric on the representation of the short-term power spectrum of a sound (lower better is better).



(a) (b)

Figure 6.4: Left: the results from Svendsen et al. [2018]. Right: illustrative data used in Svendsen et al. [2018], taken from Aires et al. [2002]. Clearly such data is poorly modelled by a single layer GP due to the heteroscedastic noise and non-stationarity, but is more appropriate for a deep model.

Chapter 7

Conclusion

This thesis has presented a number of advances in Deep Gaussian process modelling and inference. On the modelling side, Chapter 5 has introduced the ResNet He et al. [2016] style approach using the identity mean function and inner layers that match the input dimension. This allows the model to be interpreted as a form of input warping. This approach allows the DGP to be used effectively in a regression context. Chapter 5 introduced latent variable to the model, following Damianou and Lawrence [2013b], allowing the model to produce non-Gaussian marginal densities. This was shown to be highly effective in real world-tasks. On the inference side, this thesis has proposed a form of variational inference (Chapter 2), a development of the natural gradient method (Chapter 3), and an importance weighted scheme for latent variable models (Chapter 5).

The remained of the chapter considers ideas for further research, and discusses some of the limitations of the ideas presented in this thesis.

Feature Learning All the models in this thesis could be interpreted as a single layer Gaussian process with 'input warping', rather than 'feature learning'. The distinction between feature learning and input warping is somewhat arbitrary. The design of the models with mean functions and inner layer dimensions that match in the input was specifically to encourage the input warping interpretation, and so far no serious attempt has been made to identify 'features' in the inner layers. The rectangles task of Chapter 2 was specifically designed to distinguish between models that learn features and those that do not, however, and the deep model does indeed improve upon the single layer model. It is an open question to what task would conclusively demonstrate feature learning, other than image tasks where the initial

layers can be interpreted as filters. Addressing this issue is likely to be the subject of further research.

High Dimensions This thesis has only considered low to medium dimensional problems (a few hundred at the maximum, and general fewer than 20 input dimensions). There are two reasons for this: computational tractability and the difficultly of defining a meaningful covariance function in high dimensions. One way around the computational problem would be use to the linear approach employed in Chapter 4. While this scales reasonably well in dimension, the prior is increasingly degenerate in the high dimensional space, which may be inappropriate. Certainly the manifold of images is of much lower dimension than the space of all possible pixel states, but the manifold is unlikely to be linear and imposing such a linear structure may be extremely restrictive. It remains an open problem how to place probability distributions on very high dimensional spaces without extremely restrictive assumptions. For image data, in particular, the Bayesian paradigm is of questionable value as sensible priors expressible as explicit probability distributions for images are not known.

Incorporation of Prior Knowledge This thesis has many times hailed the possibility of incorporating domain specific prior knowledge into the deep GP model, but there has been no serious attempt to do this and the applications have 'black-box'. See the start of Chapter 1 for more discussion on this point. Combining the deep GP with strong prior knowledge is likely to be a fruitful avenue for potential research. Beyond the first layer, however, it is not at all clear how to do this. The inner layers are not easy to interpret, which brings into question the validity of the Bayesian paradigm. This affliction is shared by the entire sub-field of Bayesian deep learning.

Probabilistic Programming A very exciting potential development is to combine the ideas in this thesis with probabilistic programming. Probabilistic programming Gordon et al. [2014] enables a complete specification of a probabilistic model through computer code, and typically provides tools for inference that reduce the burden on the practitioner. A framework based on tensorflow [Tran et al., 2017] has recently incorporated a GP layer. Tran et al. [2018] demonstrates how to use this framework to produce the inference presented in Chapter 2. This approach allows the easy incorporation of other components such as Bayesian neural networks. Frameworks that enable the easy prototyping of models and inference schemes are likely to see the greater adoption of methods based on deep Gaussian processes.

Better Inference Throughout this thesis inference has used mean-field approximations between layers. More recent work [Havasi et al., 2018b] has relaxed this assumption, using stochastic gradient HMC. It remains an open question of how accurate this inference really is in practice (HMC is still approximate given finite computation). It may well be that the inaccuracies arising from the inference approximations are severely inhibiting performance. On the other hand, it may be the case that perfect inference would produce very similar results. It is not clear therefore

whether it is worth expending effort improving inference or modelling. Anecdotally, it appears that the errors bars of the DGP posterior are not at all well-calibrated with respect to true samples from the prior, though this is only easy to assess in 1D models where the modelling assumptions are particularly problematic.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Man, R. Monga, S. Moore, D. Murray, J. Shlens, B. Steiner, I. Sutskever, P. Tucker, V. Vanhoucke, V. Vasudevan, O. Vinyals, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2015.

R. P. Adams and O. Stegle. Gaussian process product models for nonparametric nonstationarity. *International Conference on Machine Learning*, 2008.

R. P. Adams, I. Murray, and D. J. MacKay. Nonparametric Bayesian density modeling with Gaussian processes. *arXiv:0912.4896*, 2009.

F. Aires, A. Chedin, N. A. Scott, and W. B. Rossow. A regularized neural net approach for retrieval of atmospheric and surface temperatures with the iasi instrument. *Journal of Applied Meteorology*, 2002.

M. Al-Shedivat, A. G. Wilson, Y. Saatchi, Z. Hu, and E. P. Xing. Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 2017.

M. A. Álvarez and N. D. Lawrence. Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 2011.

M. A. Alvarez, L. Rosasco, N. D. Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 2012.

M. A. Álvarez, W. O. Ward, and C. Guarnizo. Non-linear process convolutions for multi-output Gaussian processes. *Artificial Intelligence and Statistics*, 2019.

S.-I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 1998.

M. Arbel and A. Gretton. Kernel conditional exponential family. *Artificial Intelligence and Statistics*, 2018.

F. R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. *Advances in Neural Information Processing Systems*, 2009.

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 2014.

M. Bauer, M. van der Wilk, and C. E. Rasmussen. Understanding probabilistic sparse Gaussian process approximations. *Advances in Neural Information Processing Systems*, 2016.

C. M. Bishop. *Mixture Density Networks*. Aston University, 1994.

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

E. Bodin, N. D. Campbell, and C. H. Ek. Latent Gaussian process regression. *arXiv:1707.05534*, 2017.

E. V. Bonilla, K. M. Chai, and C. Williams. Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems*, 2008.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

P. Boyle and M. Frean. Dependent Gaussian processes. *Advances in Neural Information Processing Systems*, 2005.

J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv:1707.02476*, 2017.

L. Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 2001.

R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 1971.

W. Bruinsma and R. E. Turner. Learning causally-generated stationary time series. *arXiv:1802.08167*, 2018.

T. D. Bui and R. E. Turner. Stochastic variational inference for Gaussian process latent variable models using back constraints. *NeurIPS workshow on Black Box Learning and Inference*, 2015.

T. D. Bui, D. Hernández-Lobato, Y. Li, J. M. Hernández-Lobato, and R. E. Turner. Deep Gaussian processes for regression using approximate expectation propagation. *International Conference on Machine Learning*, 2016.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv:1509.00519*, 2015.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *Journal on Scientific Computing*, 1995.

M. Y. Byron, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in Neural Information Processing Systems*, 2009.

R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. *IEEE International Joint Conference on Neural Networks*, 2016.

E. Challis and D. Barber. Concave Gaussian variational approximations for inference in large-scale Bayesian linear models. *Artificial Intelligence and Statistics*, 2011.

Y. Cho and L. K. Saul. Kernel methods for deep learning. *Advances in Neural Information Processing Systems*, 2009.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 2005.

C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. *arXiv:1801.03558*, 2018.

K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep Gaussian processes. *International Conference on Machine Learning*, 2017.

K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González. Deep Gaussian processes for multi-fidelity modeling. *arXiv:1903.07320*, 2019.

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

Z. Dai, M. A. Álvarez, and N. Lawrence. Efficient modeling of latent information in supervised learning using Gaussian processes. *Advances in Neural Information Processing Systems*, 2017.

A. Damianou and N. Lawrence. Deep Gaussian processes. *Artificial Intelligence and Statistics*, 2013a.

A. Damianou and N. D. Lawrence. Semi-described and semi-supervised learning with Gaussian processes. *arXiv:1509.01168*, 2015.

A. C. Damianou and N. D. Lawrence. Deep Gaussian processes. *Artificial Intelligence and Statistics*, 2013b.

A. C. Damianou, M. K. Titsias, and N. D. Lawrence. Variational Gaussian process dynamical systems. *Advances in Neural Information Processing Systems*, 2011.

S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *arXiv:1605.07127*, 2016.

J. Domke and D. Sheldon. Importance weighting and varational inference. *Advances in Neural Information Processing Systems*, 2018.

M. M. Dunlop, M. Girolami, A. M. Stuart, and A. L. Teckentrup. How deep are deep Gaussian processes? *Journal of Machine Learning Research*, 2017.

V. Dutordoir. *Non-Stationary Surrogate Modeling with Deep Gaussian processes*. PhD thesis, Ghent University, 2017.

V. Dutordoir, H. Salimbeni, M. P. Deisenroth, and J. Hensman. Gaussian process conditional density estimation. *Advances in Neural Information Processing Systems*, 2018.

V. Dutordoir, M. van der Wilk, A. Artemev, M. Tomczak, and J. Hensman. Translation insensitivity for deep convolutional Gaussian processes. *arXiv:1902.05888*, 2019.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *International Conference on Machine Learning*, 2013.

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. *Artificial Intelligence and Statistics*, 2014.

D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive Gaussian processes. *Advances in Neural Information Processing Systems*, 2011.

Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. *Advances in Neural Information Processing Systems*, 2014.

Y. Gal, Y. Chen, and Z. Ghahramani. Latent Gaussian processes for distribution estimation of multivariate categorical data. *International Conference on Machine Learning*, 2015.

M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami. Conditional neural processes. *International Conference on Machine Learning*, 2018a.

M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh. Neural processes. *arXiv:1807.01622*, 2018b.

A. Garriga-Alonso, L. Aitchison, and C. E. Rasmussen. Deep convolutional networks as shallow Gaussian processes. *International Conference on Learning Representations*, 2019.

M. N. Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1998.

A. Girard, C. E. Rasmussen, J. Quinonero-Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. *Advances in Neural Information Processing Systems*, 2003.

T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber. Exponential natural evolution strategies. *Genetic and Evolutionary Computation*, pages 393–400, 2010.

A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. Probabilistic programming. *Proceedings of the on Future of Software Engineering*, 2014.

M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes. Deep Gaussian processes with decoupled inducing inputs. *arXiv:1801.02939*, 2018a.

M. Havasi, J. M. H. Lobato, and J. J. M. Fuentes. Inference in deep Gaussian processes using stochastic gradient Hamiltonian monte carlo. *Advances in Neural Information Processing Systems*, 2018b.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *The IEEE conference on computer vision and pattern recognition*, 2016.

A. Hebbal, L. Brevault, M. Balesdent, E.-G. Talbi, and N. Melab. Bayesian optimization using deep Gaussian processes. *arXiv:1905.03350*, 2019.

M. Heinonen, H. Mannerström, J. Rousu, S. Kaski, and H. Lähdesmäki. Non-stationary Gaussian process regression with Hamiltonian monte carlo. *Artificial Intelligence and Statistics*, 2016.

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

J. Hensman, M. Rattray, and N. D. Lawrence. Fast variational inference in the conjugate exponential family. *Advances in Neural Information Processing Systems*, 2012.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. *Uncertainty in Artificial Intelligence*, 2013.

J. Hensman, A. Matthews, M. Fillipone, and Z. Ghahramani. MCMC for variationally sparse Gaussian processes. *Advances in Neural Information Processing Systems*, 2015.

D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. Robust multi-class Gaussian process classification. *Advances in Neural Information Processing Systems*, 2011.

J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. *International Conference on Machine Learning*, 2015.

D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian Statistics*, 1999.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 2007.

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

A. Honkela, T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen. Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, 11(11):3235–3268, 2010.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv:1707.07328*, 2017.

M. Johnson, D. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in Neural Information Processing Systems*, 2016.

P. Jylänki, J. Vanhatalo, and A. Vehtari. Robust Gaussian process regression with a student-t likelihood. *Journal of Machine Learning Research*, 2011.

M. E. Khan, P. Baqué, F. Fleuret, and P. Fua. Kullback-Leibler proximal variational inference. *Advances in Neural Information Processing Systems*, 2015.

M. E. Khan, R. Babanezhad, W. Lin, M. Schmidt, and M. Sugiyama. Faster stochastic variational inference using proximal-gradient methods with general divergence functions. *Uncertainty in Artificial Intelligence*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2013.

J. Knoblauch, J. Jewson, and T. Damoulas. Generalized variational inference. *arXiv:1904.02063*, 2019.

D. A. Knowles and T. Minka. Non-conjugate variational message passing for multinomial and binary regression. *Advances in Neural Information Processing Systems*, 2011.

A. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. 1933.

T. Koriyama and T. Kobayashi. Statistical parametric speech synthesis using deep Gaussian processes. *IEEE Transactions on Audio, Speech, and Language Processing*, 2019.

K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone. AutoGP: Exploring the capabilities and limitations of Gaussian process models. *arXiv:1610.05392*, 2016a.

K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone. Autogp: Exploring the capabilities and limitations of Gaussian process models. *arXiv:1610.05392*, 2016b.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2017.

V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv:1807.00263*, 2018.

V. Kumar, V. Singh, P. Srijith, and A. Damianou. Deep Gaussian processes with convolutional kernels. *arXiv:1806.01655*, 2018.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *International Conference on Machine Learning*, 2007.

N. D. Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. *Advances in Neural Information Processing Systems*, 2004.

N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. *International Conference on Machine Learning*, 2007.

N. D. Lawrence and J. Quiñonero-Candela. Local Distance Preservation in the GP-LVM through Back Constraints. *International Conference on Machine Learning*, 2006.

M. Lázaro-Gredilla. Bayesian warped Gaussian processes. *Advances in Neural Information Processing Systems*, 2012.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as Gaussian processes. *International Conference on Learning Representations*, 2018.

H. W. Lin, M. Tegmark, and D. Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 2017.

J. R. Lloyd, D. K. Duvenaud, R. B. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. *The AAAI Conference on Artificial Intelligence*, 2014.

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? A large-scale study. *Advances in Neural Information Processing Systems*, 2018.

S. Ma and M. Belkin. Diving into the shallows: A computational perspective on large-scale shallow learning. *Advances in Neural Information Processing Systems*, 2017.

D. J. MacKay. *Bayesian modelling and neural networks*. PhD thesis, California Institute of Technology, 1991.

D. J. MacKay. Hyperparameters: optimise or integrate out. *Maximum entropy and Bayesian methods*, 1994.

D. J. MacKay. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 1998.

M. Mahsereci and P. Hennig. Probabilistic line searches for stochastic optimization. *Advances in Neural Information Processing Systems*, pages 181–189, 2015.

L. Malagò and G. Pistone. Information geometry of the Gaussian distribution in view of stochastic optimization. *Foundations of Genetic Algorithms*, 2015.

G. Marcus. Deep learning: A critical appraisal. *arXiv:1801.00631*, 2018.

J. Martens. New insights and perspectives on the natural gradient method. *arXiv:1412.1193*, 2014.

A. Matthews, J. Hensman, T. Richard, and Z. Ghahramani. On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes. *Artificial Intelligence and Statistics*, 2016a.

A. G. d. G. Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

A. G. d. G. Matthews, J. Hensman, R. E. Turner, and Z. Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *International Conference on Artificial Intelligence and Statistics*, 2016b.

A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 2017.

A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *International Conference on Learning Representations*, 2018.

C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence. Recurrent Gaussian processes. *International Conference on Learning Representations*, 2016.

K. Monterrubio-Gómez, L. Roininen, S. Wade, T. Damoulas, and M. Girolami. Posterior inference for sparse hierarchical non-stationary models. *arXiv:1804.01431*, 2018.

I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. *Advances in Neural Information Processing Systems*, 2010.

R. M. Neal. *Bayesian learning for neural networks*. Springer Science, 1996.

R. M. Neal. Regression and classification using Gaussian process priors. *Bayesian Statistics*, 1998.

A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. Deep Bayesian convolutional networks with many channels are Gaussian processes. *International Conference on Learning Representations*, 2019.

A. O'Hagan. Bayes–Hermite quadrature. *Journal of statistical planning and inference*, 1991.

A. O'Hagan, J. Bernardo, J. Berger, A. Dawid, A. Smith, et al. Uncertainty analysis and other inference tools for complex computer codes. *Bayesian Statistics*, 1998.

M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. *Information Processing in Sensor Networks*, 2008.

C. J. Paciorek. *Nonstationary Gaussian processes for regression and spatial modelling*. PhD thesis, Carnegie Mellon University, 2003.

C. J. Paciorek and M. J. Schervish. Nonstationary covariance functions for Gaussian process regression. *Advances in Neural Information Processing Systems*, 2004.

Y.-J. Park, P. M. Tagade, and H.-L. Choi. Deep Gaussian process-based Bayesian inference for contaminant source localization. *arXiv:1806.08069*, 2018.

H. Peng, S. Zhe, and Y. Qi. Asynchronous Distributed Variational Gaussian Processes. *arXiv:1704.06735*, 2017.

T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. Tighter variational bounds are not necessarily better. *International Conference on Machine Learning*, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

S. Remes, M. Heinonen, and S. Kaski. Non-stationary spectral kernels. *Advances in Neural Information Processing Systems*, 2017.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *International Conference on Machine Learning*, 2014.

S. Sæmundsson, K. Hofmann, and M. P. Deisenroth. Meta reinforcement learning with latent variable Gaussian processes. *Uncertainty in Artificial Intelligence*, 2018.

H. Salimbeni and M. Deisenroth. Deeply non-stationary Gaussian processes. *NeurIPS Workshop on Bayesian Deep Learning*, 2017a.

H. Salimbeni and M. Deisenroth. Deep Gaussian processes with importance weighted variational inference. *International Conference on Machine Learning*, 2019.

H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, 2017b.

H. Salimbeni, C.-A. Cheng, B. Boots, and M. Deisenroth. Orthogonally decoupled variational Gaussian processes. *Advances in Neural Information Processing Systems*, 2018a.

H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. *Artificial Intelligence and Statistics*, 2018b.

Y.-L. K. Samo and S. Roberts. Generalized spectral kernels. *arXiv:1506.02236*, 2015.

M.-A. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 2001.

T. Sato, G. Hütsi, and K. Yamamoto. Deconvolution of window effect in galaxy power spectrum analysis. *Progress of theoretical physics*, 2011.

M. Seeger, Y.-W. Teh, and M. Jordan. Semiparametric latent factor models. *Workshop on Artificial Intelligence and Statistics*, 2005.

A. Shah, A. Wilson, and Z. Ghahramani. Student-t processes as alternatives to Gaussian processes. *Artificial Intelligence and Statistics*, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

E. Snelson, Z. Ghahramani, and C. E. Rasmussen. Warped Gaussian processes. *Advances in Neural Information Processing Systems*, 2004.

J. Snoek, K. Swersky, R. Zemel, and R. Adams. Input warping for Bayesian optimization of non-stationary functions. *International Conference on Machine Learning*, 2014.

K. Sohn, H. Lee, and X. Yan. Learning Structured Output Representation using Deep Conditional Generative Models. *Advances in Neural Information Processing Systems*, 2015a.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems*, 2015b.

A. Solin, M. Kok, N. Wahlström, T. B. Schön, and S. Särkkä. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on Robotics*, 2018.

B. Sriperumbudur, K. Fukumizu, A. Gretton, A. Hyvärinen, and R. Kumar. Density estimation in infinite dimensional exponential families. *Journal of Machine Learning Research*, 2017.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, 1974.

S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Grosse. Differentiable compositional kernel learning for Gaussian processes. *International Conference on Machine Learning*, 2018.

Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. *Genetic and Evolutionary Computation*, pages 539–546, 2009.

S. Sundararajan and S. S. Keerthi. Predictive approaches for choosing hyperparameters in Gaussian processes. *Advances in Neural Information Processing Systems*, 2000.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning*, 2013.

D. H. Svendsen, P. Morales-Álvarez, R. Molina, and G. Camps-Valls. Deep Gaussian processes for geophysical parameter retrieval. *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018.

A. Svensson, J. Dahlin, and T. B. Schön. Marginalizing Gaussian process hyperparameters using sequential monte carlo. *Computational Advances in Multi-Sensor Adaptive Processing*, 2015.

Y. Tang. Deep learning using linear support vector machines. *Internation Confernece on Machine Learning*, 2013.

M. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. *Artificial Intelligence and Statistics*, 2009.

M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. *Artificial Intelligence and Statistics*, 2010a.

M. Titsias and M. Lázaro-Gredilla. Variational inference for Mahalanobis distance metrics in Gaussian process regression. *Advances in Neural Information Processing Systems*, 2013.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. *Artificial Intelligence and Statistics*, 2010b.

M. K. Titsias, N. Lawrence, and M. Rattray. Markov chain monte carlo algorithms for Gaussian processes. *Inference and Estimation in Probabilistic Time-Series Models*, 2008.

F. Tobar, T. D. Bui, and R. E. Turner. Learning stationary time series using Gaussian processes with nonparametric kernels. *Advances in Neural Information Processing Systems*, 2015.

J. Townsend, D. Duvenaud, and J. Matthew. Autograd issue 175. `https://github.com/HIPS/autograd/pull/175`, 2017.

B. D. Tracey and D. Wolpert. Upgrading from Gaussian processes to student'st processes. *AIAA Non-Deterministic Approaches Conference*, 2018.

D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei. Deep probabilistic programming. *arXiv:1701.03757*, 2017.

D. Tran, D. Mike, M. van der Wilk, and D. Hafner. Bayesian layers: A module for neural network uncertainty. *arXiv:1812.03973*, 2018.

G. Tucker, D. Lawson, S. Gu, and C. J. Maddison. Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives. *International Conference on Learning Representations*, 2019.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, 2011.

K. Vafa. *Training and Inference for Deep Gaussian Processes*. PhD thesis, Harvard College, 2016.

A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

A. W. van der Vaart, J. H. van Zanten, et al. Rates of contraction of posterior distributions based on Gaussian process priors. *The Annals of Statistics*, 2008.

C. Wang and R. Neal. Gaussian process regression with heteroscedastic or non-Gaussian residuals. *arXiv:1212.6246*, 2012.

Y. Wang, M. Brubaker, B. Chaib-Draa, and R. Urtasun. Sequential inference for deep Gaussian process. *Artificial Intelligence and Statistics*, 2016.

C. K. Williams. Computing with infinite networks. *Advances in Neural Information Processing Systems*, 1997.

A. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning*, 2013.

A. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). *International Conference on Machine Learning*, 2015.

A. G. Wilson, D. A. Knowles, and Z. Ghahramani. Gaussian process regression networks. *arXiv:1110.4411*, 2011.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. *Artificial Intelligence and Statistics*, 2016.

Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the Quantitative Analysis of Decoder-based Generative Models. *International Conference on Learning Representations*, 2016a.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016b.

Y. Yao, A. Vehtari, D. Simpson, and A. Gelman. Yes, but did it work?: Evaluating variational inference. *arXiv:1802.02538*, 2018.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

Q. Zhang and S. Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 2018.