Scalable Gaussian process inference using variational methods

Alexander Graeme de Garis Matthews

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Darwin College

September 2016

Declaration

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

Alexander Graeme de Garis Matthews
September 2016

Acknowledgements

First I would like to thank my supervisor Zoubin Ghahramani. When I moved back to Cambridge I hoped that we would write about variational inference and indeed that was how it turned out. It has been a privilege to work so closely with him. He brings the best out in the people around him and has always supported my development. I am very much looking forward to continuing to work with him next year.

My work with James Hensman began when he visited Zoubin in the summer of 2014. Since then we have worked together on the four related projects that eventually became this thesis. This working relationship was, in my view, an extremely productive one. I feel fortunate to have worked with James, who I consider to be a world expert in his area.

I wish to thank my other collaborators. Richard Turner is a great person to work with. His door is always open and indeed on one visit to his office I had the debate that set chapter 3 in motion. I learnt a great deal about MCMC working with Maurizio Filippone on chapter 5. I will remember our marathon poster session at NIPS for a long time.

I would like to thank Mark van der Wilk, Tom Nickson, and all the GPflow contributors who are listed on the project web page, along with Rasmus Munk Larsen who reviewed my contribution to TensorFlow.

David MacKay, who sadly passed away this year, was an inspiration to me. Taking his now famous ITALA course changed the course of my career. He will be greatly missed.

It is important that I acknowledge the thriving intellectual environment that is CBL in Cambridge. I would like to particularly highlight important conversations with Thang Bui and James Lloyd.

This PhD was mostly funded by the UK Engineering and Physics Research Council. Thank you to them for their support.

I am grateful to Carl Rasmussen and Stephen Roberts for examining my thesis. Any remaining inaccuracies are of course my own.

Finally, on a personal note, I would like to thank my amazing wife Rachel and my family, particularly my parents who gave me every opportunity.

More detail on the mapping between the conference papers I wrote with my coauthors and the chapters of this thesis can be found in the *publications* section.

Abstract

Gaussian processes can be used as priors on functions. The need for a flexible, principled, probabilistic model of functional relations is common in practice. Consequently, such an approach is demonstrably useful in a large variety of applications.

Two challenges of Gaussian process modelling are often encountered. These are dealing with the adverse scaling with the number of data points and the lack of closed form posteriors when the likelihood is non-Gaussian. In this thesis, we study variational inference as a framework for meeting these challenges.

An introductory chapter motivates the use of stochastic processes as priors, with a particular focus on Gaussian process modelling. A section on variational inference reviews the general definition of Kullback-Leibler divergence. The concept of prior conditional matching that is used throughout the thesis is contrasted to classical approaches to obtaining tractable variational approximating families.

Various theoretical issues arising from the application of variational inference to the infinite dimensional Gaussian process setting are settled decisively. From this theory we are able to give a new argument for existing approaches to variational regression that settles debate about their applicability. This view on these methods justifies the principled extensions found in the rest of the work.

The case of scalable Gaussian process classification is studied, both for its own merits and as a case study for non-Gaussian likelihoods in general. Using the resulting algorithms we find credible results on datasets of a scale and complexity that was not possible before our work. An extension to include Bayesian priors on model hyperparameters is studied alongside a new inference method that combines the benefits of variational sparsity and MCMC methods. The utility of such an approach is shown on a variety of example modelling tasks.

We describe GPflow, a new Gaussian process software library that uses TensorFlow. Implementations of the variational algorithms discussed in the rest of the thesis are included as part of the software. We discuss the benefits of GPflow when compared to other similar software. Increased computational speed is demonstrated in relevant, timed, experimental comparisons.

Publications

Chapter 3 expands on:

Alexander G de G Matthews, James Hensman, Richard E. Turner, and Zoubin Ghahramani. *On Sparse Variational methods and the Kullback-Leibler divergence between stochastic processes.* In 19th International Conference on Artificial Intelligence and Statistics, Cadiz, Spain, May 2016.

The key theorems are now stated and proved formally. Sections 3.3.4 and 3.5 are novel for this thesis. Section 3.4 has been rewritten.

Chapter 4 and 5 correspond respectively to:

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. *Scalable Variational Gaussian Process Classification.* In 18th International Conference on Artificial Intelligence and Statistics, San Diego, California, USA, May 2015.

James Hensman, Alexander G de G Matthews, Maurizio Filippone, and Zoubin Ghahramani. *MCMC for Variationally Sparse Gaussian Processes.* In Advances in Neural Information Processing Systems 28, Montreal, Canada, December 2015.

I have personally rewritten these papers for this thesis. In particular, it was felt that ‘Scalable Variational Gaussian Process Classification’ could be made significantly clearer in light of the theoretical innovations of Chapter 3. There are new experiments and some experiments have been moved. Chapter 5 now highlights the concept of variational potentials that did not appear in the conference paper. Both of these chapters benefit from no longer having to meet length constraints and from being presented harmonized with the other chapters.

Chapter 6 describes an open source software project available at:

https://github.com/GPflow/gpflow

This is the first time the design and motivation of the project have been discussed in a paper. An up to date list of contributors can be found on the web page.

Table of contents

List of figures

List of tables

Chapter 1

Stochastic processes as priors

A stochastic process is a distribution on a potentially infinite set of random variables. A Gaussian process is a type of stochastic process. The study of the use of stochastic processes as Bayesian priors is called Bayesian nonparametrics.

The aims of this chapter are to understand and motivate the use of Gaussian processes as priors; to understand the challenges that arise when using them in practice, and to set the stage for the solutions we advocate in the rest of this thesis. Our aims have some overlap with previous literature introducing Gaussian processes, for instance the more general text of Rasmussen and Williams (2006). Our exposition emphasizes the mathematics of the Kolmogorov extension theorem and other central ideas that will be needed during this thesis. Along with the variational inference concepts introduced in chapter 2, these are the key foundations to prepare the reader for the main theoretical contributions of the thesis, which begin in earnest in chapter 3.

In detail, we start with the relatively technical section 1.1, explaining the Kolmogorov extension theorem and its application to Gaussian processes. Next, in the more conceptual section 1.2.1, we motivate the use of stochastic processes as priors. Section 1.2.2 then discusses some technical issues that arise when applying Bayes' theorem in this context. Gaussian process modelling is then introduced in sections 1.2.3, 1.2.4 and 1.2.5. We finish by highlighting two common challenges of using Gaussian processes in this way which constitute major themes of this thesis (section 1.2.6).

We do not discuss the standard formulation of probability in terms of measure theory, instead leaving this to existing texts (Billingsley, 1995; Capinski and Kopp, 2004). Measure theory is used in the majority of this chapter. However, sections 1.2.1, 1.2.4, 1.2.5 and 1.2.6 do not require measure theory and should be of standalone interest.

1.1 Stochastic processes

In this section we explain one route to formulate probability distributions on infinite dimensional spaces. We will concentrate on the statement of the Kolmogorov extension theorem without proof; the application of the theorem to Gaussian processes; and some limitations of the theorem.

1.1.1 The Kolmogorov extension theorem

The Kolmogorov extension theorem concerns the existence of probability distributions for an infinite dimensional object based on a consistency relation for finite dimensional marginal distributions of that object. In what follows the infinite dimensional objects in question will be functions f. For our exposition of the theorem we were influenced by those of Sengupta (2014) and Billingsley (1995) and our notation is influenced in part by the former.

We will first establish some notation. Consider a function f mapping an index set X to the set of real numbers $f : X \mapsto \mathbb{R}$. The index set X may be finite, countably infinite or uncountably infinite. Entirely equivalently we may write $f \in \mathbb{R}^X$ and this notation leads us to consider functions as vectors with elements indexed by members of X. In the case where X is infinite this view corresponds to an infinite dimensional vector. Again equivalently we can use sequence notation $(f(x))_{x \in X}$. Pursuing this direction, we also define set indexing of the function. If $S \subseteq X$ is some subset of the index set, then $f_S := (f(x))_{x \in S}$ and we may straightforwardly extend this definition to single elements of the index set $f_x := f_{\{x\}}$. The whole function f may be denoted f_X and we will make use of both versions of this. When the latter is chosen it will often be to emphasize this dependance on the whole index set.

Consider a projection map $\pi_{U \to V}$ that for $V \subset U \subseteq X$ has the following property:

$$\pi_{U \to V} : \mathbb{R}^U \mapsto \mathbb{R}^V : (f(x))_{x \in U} \mapsto (f(x))_{x \in V} \tag{1.1}$$

We denote by $\mathcal{B}(\mathbb{R}^D)$ the Borel σ-algebra of a space $\mathbb{R}^D$. A *cylinder set* is the pre-image of the projection $\pi_{X \to V}$ of a Borel set $E \in \mathcal{B}(\mathbb{R}^V)$ for finite V, denoted:

$$\pi^{-1}_{X \to V}(E) = \{f \in \mathbb{R}^X : \pi_{X \to V}(f) \in E\} \tag{1.2}$$

In other words, the cylinder set $\pi^{-1}_{X \to V}(E)$ is the set of all functions whose values on the finite set V lie within the Borel set E. We show this schematically using an example in figure 1.1. The σ-algebra on which the probability measure over functions is defined in the Kolmogorov extension theorem is the *product* σ-algebra. The product σ-algebra is the smallest σ-algebra which contains all the cylinder sets $\pi^{-1}_{X \to V}(E)$ for all finite $V \subset X$

and Borel sets E. The product σ-algebra contains many events of interest but there are some events that are potentially of interest that are not included. This will be discussed further in section 1.1.4. Having set up notations and definitions we are now ready to state the Kolmogorov extension theorem.

Theorem: Kolmogorov. *If a family of Borel probability measures, labelled by their corresponding finite index set, μ_V obeys the following consistency relation for all $V \subset X$ and all second finite index sets with $U \supset V$*

$$\mu_U(\pi^{-1}_{U \to V}(E)) = \mu_V(E) \tag{1.3}$$

then there is a unique probability measure on the product σ-algebra, μ_X with the property:

$$\mu_X(\pi^{-1}_{X \to V}(E)) = \mu_V(E) \quad \forall E \in \mathcal{B}(\mathbb{R}^V)\,. \tag{1.4}$$

The collection of measures μ_V are called the finite dimension distributions or finite dimensional marginals of the measure μ_X.

An intuitive way to think about the result is to consider it in reverse. If such a measure μ_X existed it would necessarily obey equation (1.4) for consistency reasons. The Kolmogorov theorem tells us we can go the other way. If there is a collection of measures that obey the consistency property we know that there exists a unique probability measure on the product σ-algebra which has those measures as finite dimensional marginals. Note that the result is very general and holds even if X is uncountably infinite.

1.1.2 Gaussian processes

Here we discuss the relationship between Gaussian processes and their finite dimensional marginal distributions. We denote by $\mathcal{N}\{m, \Sigma\}(\cdot)$ a multivariate Gaussian measure which is parameterized by a mean vector $m \in \mathbb{R}^D$ and a $D \times D$ positive semi-definite matrix Σ. We can define mean vectors m and matrices Σ for a finite subset $U \subset X$ using a mean function $\lambda : X \mapsto \mathbb{R}$ and kernel function $K : X \times X \mapsto \mathbb{R}$ respectively. The components of the mean vector can be obtained using the mean function:

$$m(U)_i = \lambda(x_i) \quad x_i \in U\,. \tag{1.5}$$

The components of the matrix Σ are given by

$$\Sigma(U)_{i,j} = K(x_i, x_j) \quad x_i, x_j \in U \times U\,. \tag{1.6}$$

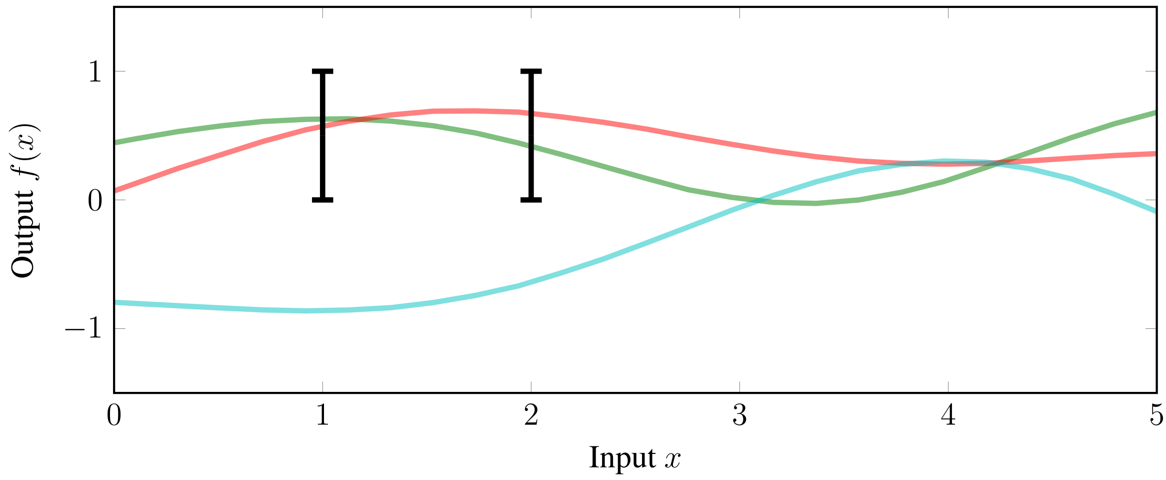

Fig. 1.1 A visualization of the cylinder sets $\pi^{-1}_{X\to V}(E)$. In this case the index set $X = [0, 5]$, the subset of the index set V is $\{1, 2\}$ and the Borel set E is $[0, 1] \times [0, 1]$. The red and green functions are members of the cylinder sets because they have function values $0 \leq f(1) \leq 1$ and $0 \leq f(2) \leq 1$. The blue function is not a member of the cylinder set in question. Although all the example functions shown are continuous, there is nothing in the definition of a cylinder set that requires this.

The defining property of a kernel function is that all such matrices be positive semi-definite. The matrices $\Sigma(U),\ U \subset X$ are referred to as *Gram matrices*.

Multivariate Gaussian measures obey a *marginalization* property. For the case of multivariate Gaussian measures specified using a mean and kernel function on an index set it can be stated in the following way. For $V \subset U \subset X$ with U, V finite

$$\mathcal{N}\left\{m(U), \Sigma(U)\right\}(\pi^{-1}_{U\to V}(E)) = \mathcal{N}\left\{m(V), \Sigma(V)\right\}(E) \quad E \in \mathcal{B}(\mathbb{R}^V)\,. \tag{1.7}$$

The accumulation of notation is rather dense here, so we restate this marginalization property in words. First we obtain a mean vector and Gram matrix using the mean and covariance functions on some finite subset U of the overall index set X. We can associate with this mean vector and Gram matrix a multivariate Gaussian measure. This multivariate Gaussian measure will be consistent under marginalization with the multivariate Gaussian measure we would have obtained had we used subset $V \subset U$ in place of U itself. Equation (1.7) tells us that the family of multivariate normal measures associated with the mean function λ and the kernel function K has precisely the consistency property necessary for the Kolmogorov extension theorem. Therefore there is a measure on the set of all functions with events in the product σ-algebra which has this family of measures as finite dimensional marginals. This probability distribution is referred to as a *Gaussian process*[1]. In the context of Gaussian processes, the kernel function is also called the *covariance function* because it gives the covariance between two of the random function values. We will use these terms interchangeably.

1.1.3 Examples

We here give two examples of covariance functions. Many more examples are discussed by Rasmussen and Williams (2006).

First let the index set be the real line $\mathbb{R}$. The following is a valid kernel for that index set:

$$K(x_1, x_2) = \sigma^2 \exp\left\{ -\frac{1}{2l^2}(x_1 - x_2)^2 \right\} \tag{1.8}$$

The kernel is commonly used and goes by a number of names: the *Radial Basis Function* (RBF) kernel, the *Gaussian* kernel, and the *Exponentiated Quadratic* kernel. The hyperparameter σ^2 controls the single variable marginal variance of the process, which is constant. The parameter l controls the length scale over which the process is correlated. The kernel is *stationary*: the implied distribution on functions is translationally invariant. Some draws from the RBF kernel are shown in figure 1.2 (top).

Second let the index set be the positive real line $[0, \infty)$. The following is a valid kernel for that index set.

$$K(x_1, x_2) = \lambda^2 \min\left(\frac{x_1}{l}, \frac{x_2}{l}\right) \tag{1.9}$$

We will call this kernel the *Wiener kernel* because it has the finite dimensional marginals of a *Wiener process*. The Wiener process (Wiener and Masani, 1976) is a mathematical model of *Brownian* motion (Brown, 1828; Einstein, 1905) and arises as the careful formulation of a continuous time random walk. The parameter $l > 0$ rescales the input parameter and the parameter $\lambda > 0$ is a scaling parameter on the output space. Some draws from the Wiener kernel are shown in figure 1.2 (bottom). The marginal variance is $\lambda^2 x$. Even if we extended the index set to the whole real line, the corresponding process would not be stationary. The process does have the *Markov property*: the future evolution of the process is conditionally independent of the past given the present state. The kernel has a certain transformation invariance property. If we take $\beta \in [0, \infty)$ and use it to create new parameters $\tilde{\lambda} = \sqrt{\beta}\lambda$ and $\tilde{l} = \beta l$ then the kernel is unchanged. These properties of the kernel mean that a close

[1] The use of the term 'process' is a historical artefact of the case where the index set X was a time like variable indexed using the real line, but the terms 'Gaussian process' and more generally 'stochastic process' are now often used in relation to any index set X and we will follow this convention.

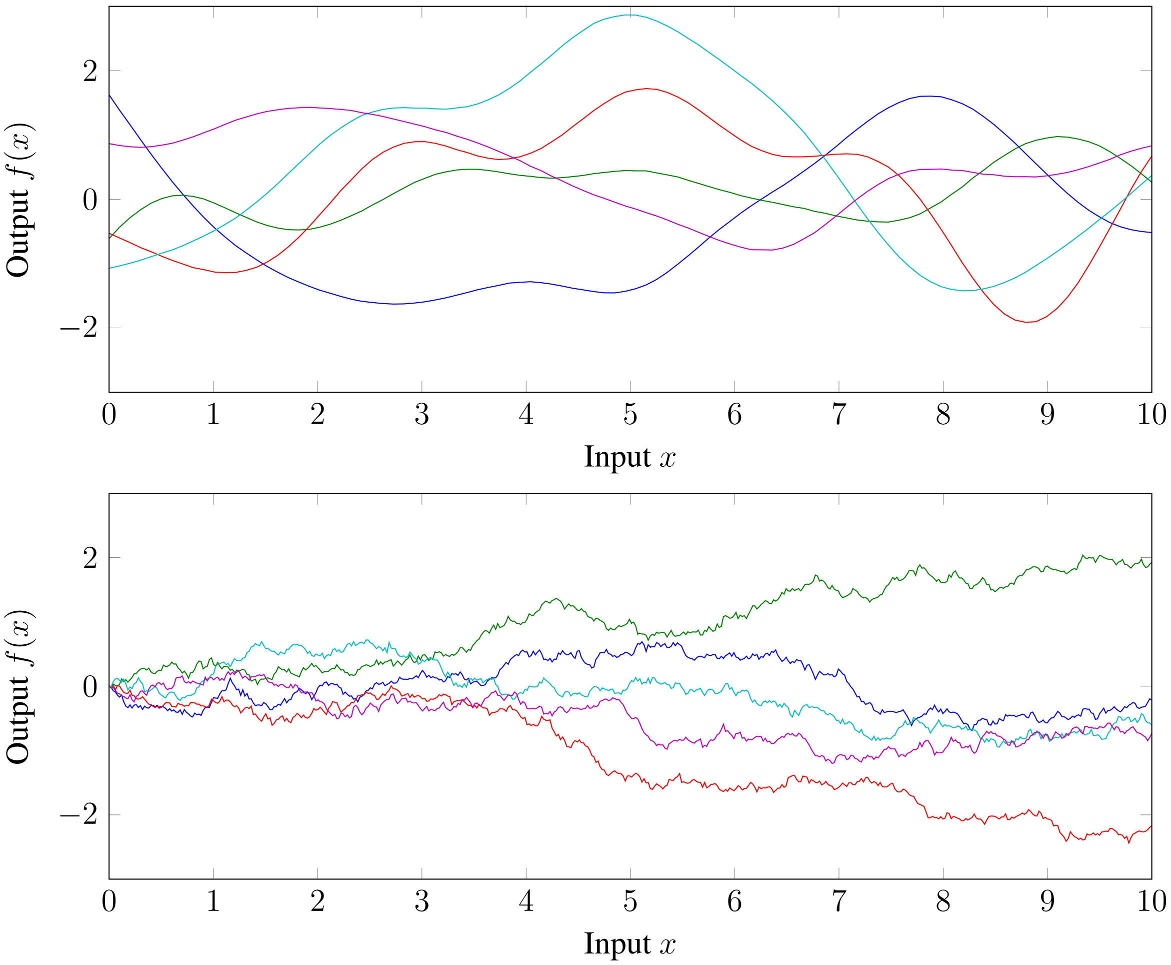

Fig. 1.2 Example draws from Gaussian processes with two different kernels. Top: Five sample functions with an RBF kernel with $l = \sigma = 1$. Bottom: Five sample functions with a Wiener kernel with $\lambda = 0.3$ and $l = 1$.

investigation of any line segment of the sample draws would reveal a ‘random fractal’ like behaviour. Although there would not be exact self similarity, the random function behaviour would look similar at the different scales.

1.1.4 Limitations of the product σ-algebra

There are certain events that one might want to consider in probabilistic modelling that are not in the product σ-algebra and are therefore not specified for functions defined using the Kolmogorov extension theorem. In this section we will explain this issue and approaches to consistently defining probabilities for the missing events. The presentation we give is influenced by those of Adler (1981) and Billingsley (1995).

Consider a simple motivating example. Imagine that we find ourselves playing the role of a physicist using the Wiener process as a model of one dimensional particle diffusion. Suppose we are interested in a physical boundary at $f(x) = c$. Perhaps there is a change in the diffusion constant here, or a membrane. We might be interested in the question: 'What is the probability that the particle has not had a position greater than or equal to c at any time up to and including time $x = T$?' As a well posed question this has a precise mathematical meaning. We are interested in the probability of the event:

$$E_T = \{f(x) : f(x) < c \ \ \forall x \in [0, T]\} \tag{1.10}$$

Taking this event, we might then hope to evaluate its probability using the Gaussian process measure μ_X resulting from the Kolmogorov extension theorem with zero mean function and a Wiener kernel. The problem that we will encounter, unfortunately, is that the event E_T is not a member of the product σ-algebra and is hence not measurable. Nor is this the only relevant example. The questions: 'What is the probability the function will be continuous?' and 'What is the probability the function will be differentiable?' also correspond to events that are not coverred by the product σ-algebra. We need some way to 'extend' the measure μ_X to a larger σ-algebra in a way that agrees with the original definition when it applies. Technically, however, this is not at all straight forward. More discussion of this point and of *separability* are given in Appendix A. Using such ideas one can find an extension of the RBF kernel Gaussian process that has sample draws which are almost surely both continuous and everywhere infinitely differentiable and a version of the Wiener kernel Gaussian process that almost surely has sample draws which are continuous but nowhere differentiable.

In this section we have drawn the reader's attention to some important caveats in the use of the Kolmogorov extension theorem, but for many cases of interest in probabilistic modelling using Gaussian processes, including the majority of cases in this thesis, the product σ-algebra will be sufficient. The exception in this work is the example extensions of the sparse variational inference scheme presented in section 3.6 where we consider random variables based on integrals of Gaussian processes.

1.2 Bayesian nonparametrics

In this section we discuss the motivation of *Bayesian nonparametrics* (BNP) and then discuss how to apply Bayes' theorem to infinite dimensional models. Finally we talk about the specific case of Gaussian processes.

1.2.1 Why Bayesian nonparametrics?

The idea of using infinite dimensional objects for priors is the core concept of Bayesian nonparametrics (BNP). The philosophy and motivation of this area has been well discussed by a number of authors (Ghosh and Ramamoorthi, 2003; Hjort et al., 2010; Orbanz and Teh, 2010). For this thesis we would particularly highlight two works because they have had a significant influence on the ideas presented. A review by Ghahramani (2012) contains an approachable discussion of the philosophy behind Bayesian nonparametric models. Schervish's take on the foundations of statistics (1995) is notable not only for providing a compelling motivation for nonparametric Bayesian thinking, but also for presenting it next to frequentist methods in a unified view. Given the excellent range of discussion on the motivation question we will only highlight some key points for the perspective of this work.

The Bayesian/non-Bayesian debate is still alive despite at least a century of discussion. This is despite a strong theoretical basis for Bayesian reasoning, for instance in the early axiomatic derivations of Cox (1946) and the work of de Finetti (1974). The central tenet is that in order to be coherent the only way to represent and reason with uncertainty is to use the rules of probability. The differences between frequentist and Bayesian paradigms are well understood through the lens of decision theory. See for instance chapter 3 of Schervish's book (1995). An additional consideration that is particularly relevant to this thesis is that of computational constraints. Although the Bayesian framework tells us exactly how to go from prior to posterior in theory, computationally this can sometimes be very difficult. A pragmatic answer, and one that is adopted in this thesis, is to provide, according to some reasonable criteria, the best approximations we can for a useful class of models. The more general question of how to act rationally in the presence of computational constraints from a Bayesian perspective is still, in our opinion, in its infancy.

The nonparametric ingredient is perhaps best motivated by comparison with non-Bayesian nonparametric methods such as K-nearest neighbour classification, which is in fact how such methods came about historically. The K-nearest neigbour classification algorithm does not rely on a parametric expression of the discrimination function. Instead it ultimately relies on the fact that with enough data a given test example should be close to other examples of its true class as measured by the chosen distance metric. This weaker assumption about the discrimination function means that the K-nearest neighbour algorithm will perform well assymptotically in a variety of situations. There are however some criticisms of such a classical nonparametric model. It is difficult to incorporate a principled treatment of predictive uncertainty, to generalize the algorithm, or to incorporate prior knowledge. All of these issues motivate a probabilistic analogue of nonparametric methods. It is worth noting that in the probabilistic domain the label nonparametric needs to be treated with care. Bayesian 'nonparametric' models actually have infinitely many parameters which are given a Bayesian prior. For instance, by using a Gaussian process we can place a prior on the function values over an infinite index set. This is often much less constraining in terms of model flexibility than a parametric Bayesian approach. Again, the implications of limited computational resources to the nonparametric argument have to be considered. This is true even in the case of the K-nearest neighbour classifier. With the simplest implementation, as new data arrives we have to store all of it and the cost of prediction grows as we need to compare against many different examples. It seems desirable to *compress* or simplify the prediction process with as little distortion as possible. This view is certainly relevant to what follows. The inducing point approximations we study use *pseudo data* that is tuned to bring the approximation close to the full nonparametric posterior.

1.2.2 Bayes' theorem for infinite dimensional models

In BNP the priors used are sufficiently complex mathematically that we will often need to use measure theory. The equations of Bayesian inference can look somewhat different at this level of abstraction and for this reason we review them here. To do this we will broadly follow the presentation of Schervish (1995) which is a good starting point for more detail.

In this thesis we will usually be interested in regression models or more generally discriminative models where we are presented with input and output data and wish to represent the mapping between the two. Adapting Bayes' theorem to this purpose from Schervish's expression for a dominated model (Schervish, 1995) we obtain:

$$\frac{d\hat{P}}{dP}(f) = \frac{p_X(Y|f)}{p(Y)} \,. \tag{1.11}$$

Here P is the prior measure over the functional mapping f from inputs to outputs, $\hat{P}$ is the posterior over the functional mapping and the derivative on the left hand side is a Radon-Nikodym derivative. Y is the output data. The function $p_X(Y|\cdot)$ is the *likelihood* which maps to the non-negative reals. $p(Y) := \int p_X(Y|f)dP(f)$ is the *marginal likelihood*. The dominating measure in this expression is the prior. If we have a σ-finite third measure μ which dominates the prior then the chain rule for Radon-Nikodym derivatives implies that

$$\frac{d\hat{P}}{d\mu}(f) = \frac{d\hat{P}}{dP}(f)\frac{dP}{d\mu}(f) \tag{1.12}$$

$$= \frac{p_X(Y|f)}{p(Y)}\frac{dP}{d\mu}(f)\,. \tag{1.13}$$

In the case where f is finite dimensional and the prior has density with respect to Lebesgue measure m we can take $\mu = m$ and recover the familiar expression for Bayes' theorem with densities. As discussed in section 3.3.1, there is no sensible infinite dimensional Lebesgue measure. We therefore need to use equation (1.11) when working with dominated infinite dimensional models. By the defining property of Radon-Nikodym derivatives, we may entirely equivalently write equation (1.11) in integral form, whereupon it gives the posterior probability of an event E.

$$\hat{P}(E) = \int_E \frac{p_X(Y|f)}{p(Y)} dP(f) \tag{1.14}$$

In many cases the likelihood only actually depends on the function values f_D at a finite subset $D \subseteq X$ of the index set, with $|D| = N$. We choose the symbol D because this subset often corresponds to the data inputs. f_D is a finite vector as opposed to the potentially infinite function f. This can be expressed as:

$$p_X(Y|f) = p(Y|\pi_D(f)) \tag{1.15}$$

where $\pi_D : \mathbb{R}^X \mapsto \mathbb{R}^D$ is a projection as before and $p(Y|\cdot)$ is a likelihood function that only depends only on f_D, a finite vector[2]. Under this assumption the discriminative Bayes' theorem becomes

$$\frac{d\hat{P}}{dP}(f) = \frac{p(Y|\pi_D(f))}{p(Y)} \tag{1.16}$$

where, in this case, the marginal likelihood has a simplified expression

$$p(Y) = \int p(Y|f_D)dP_D(f_D)\,. \tag{1.17}$$

[2] We distinguish the likelihood $p_X(Y|f)$ for the full function f_X from the likelihood $p(Y|f_D)$ for the data function f_D only. These are different mathematical objects. The notation for $p(Y|f_D)$ has been chosen so as to match standard notation as closely as possible.

Here P_D is the prior marginal distribution of the data function values. In general μ_C with $C \subset X$ will denote the marginal distribution of measure μ for the function values f_C. We could also, if required, find an integral representation for equation (1.16).

Now we consider a special case of equation (1.16). Define a finite dimensional vector f_* of predictive function points that are not data points. Consequently $f_{*\cup D}$ is the finite vector corresponding to the function at the predictive points and the data points. If we assume that the subset E has the specific structure of a cylinder set $\pi^{-1}_{X\to *\cup D}(C)$ then

$$\hat{P}(\pi^{-1}_{X\to *\cup D}(C)) = \hat{P}_{*\cup D}(C) = \int_{\pi^{-1}_{X\to *\cup D}(C)} \frac{p(Y|\pi_D(f))}{p(Y)} dP(f) \tag{1.18}$$

$$= \int_C \frac{p(Y|\pi_{D\cup *\to D}(f_{D\cup *}))}{p(Y)} dP_{D\cup *}(f_{D\cup *})\,. \tag{1.19}$$

This probability therefore only depends on a finite dimensional integral.

1.2.3 Using Gaussian processes as priors

We now take the case where the prior measure P is a Gaussian process with zero mean and covariance function K. Here we will assume that likelihood only depends on the function f_D. Under this assumption the discriminative Bayes' theorem of equation (1.16) applies. We assume in this section that the Gaussian prior has nondegenerate marginal distributions which means the prior finite dimensional marginals have density with respect to Lebesgue measure. The posterior process $\hat{P}$ is uniquely specified on the product σ-algebra by its finite dimensional marginals, which also have density with respect to Lebesgue measure. The joint posterior density of f_D and f_* is given by

$$p(f_D, f_*|Y) = \frac{p(Y|f_D)p(f_*, f_D)}{p(Y)} \tag{1.20}$$

$$= \frac{p(Y|f_D)p(f_D)p(f_*|f_D)}{p(Y)}\,. \tag{1.21}$$

We have introduced some new notation. $p(f_D)$ and $p(f_*, f_D)$ are the densities with respect to Lebesgue measure of the Gaussian process prior marginal distributions P_D and $P_{D\cup *}$ respectively. $p(f_D, f_*|Y)$ is the density of the posterior marginal distribution $\hat{P}_{D\cup *}$ with respect to Lebesgue measure. $p(f_*|f_D)$ is the prior conditional density of f_* conditioned on f_D.

It is informative to understand the relationship between equations (1.18) and (1.20). Equation (1.18), which applies to cases other than Gaussian processes, is an integrated version of equation (1.20) and it does not assume densities with respect to Lebesgue measure.

We use the notation $K_{A,B}$ to denote a matrix whose elements correspond to the kernel function evaluated on each pair of elements in the set $A \times B$. For example, the Gram matrix $\Sigma(U)$ in equation (1.6) would be written in this shorthand as $K_{U,U}$. As the prior density of the marginal distribution for f_D, the term $p(f_D)$ is a normal density given by

$$p(f_D) = \mathcal{N}(f_D|0, K_{D,D})\,. \tag{1.22}$$

The prior conditional density $p(f_*|f_D)$ is also normal

$$p(f_*|f_D) = \mathcal{N}(f_*|K_{*,D}K_{D,D}^{-1}f_D, K_{*,*} - K_{*,D}K_{D,D}^{-1}K_{D,*})\,. \tag{1.23}$$

In the special case where the likelihood $p(Y|f_D)$ is a Gaussian density $\mathcal{N}(Y|f_D, \epsilon^2 I)$ centred on f_D and with isotropic covariance matrix $\epsilon^2 I$, the posterior distrubtion is again normal, which can be very convenient in applications. The density in this case is given by

$$p(f_D, f_*|Y) = \mathcal{N}(f_D, f_*|K_{D\cup *,D}\Delta^{-1}Y, K_{D\cup *,D\cup *} - K_{D\cup *,D}\Delta^{-1}K_{D,D\cup *}) \tag{1.24}$$

where $\Delta = K_{D,D} + \epsilon^2 I$. If the likelihood is not Gaussian there is not usually a closed form expression for the posterior density. This is a consequence of the lack of a closed form for the marginal likelihood integral $p(Y)$.

1.2.4 Hyperparameter selection

A given Gaussian process model has *hyperparameters* associated with it. For instance with an RBF kernel and a Gaussian likelihood the hyperparameters would be the noise variance ϵ^2, the signal variance σ^2 and the kernel lengthscale l. It will often be difficult to choose fixed hyperparameters in practice and different values can give very different posteriors. Therefore a hyperparameter selection method is often necessary. The strict Bayesian approach here is to put a prior on the hyperparameters. This usually introduces additional intractability. Approximations which handle the fully Bayesian case are discussed in chapter 5. It is often the case in practice that a point estimate is chosen by maximizing $p(Y)$ as a function of the hyperparameters. This can be viewed as a delta function approximation to the full Bayesian solution. This is because, for many standard cases, as the number of data points increases the posterior over hyperparameters concentrates in a tight ball around the maximum likelihood solution (Rasmussen and Williams, 2006). This idea has also been discussed under the name 'Empirical Bayes' and 'Type-II maximum likelihood' and has been shown to stand up well under a frequentist analysis (Murphy, 2012, Chapter 5) (Carlin and Louis, 2010; Efron, 2010).

1.2.5 Examples of Gaussian process regression

We now show examples of Gaussian process priors applied to a simple example dataset. A readily visualized example dataset was supplied by Snelson and Ghahramani (2005) in their paper and (with apologies to Ghahramani), we will adopt the commonly used name 'Snelson dataset'. We use a Gaussian likelihood with shared noise variance ϵ^2. We compare regression with an RBF and Wiener kernel. The hyperparameters are chosen using type-II maximum likelihood. The results are shown in figure 1.3. For the RBF kernel the posterior samples are very smooth. The predictions revert to the stationary prior away from the data. For the Wiener kernel the posterior samples are very rough. Away from the data the posterior predictions are more influenced by the nonstationary prior which has 0 variance at $x = 0$. As such, for many regression problems the Wiener kernel would be an unnatural choice of prior. Note in both cases the output space predictions are more uncertain than the function space predictions, because they account for the noise variance in the likelihood.

1.2.6 Two challenges of Gaussian process modelling

We highlight two issues that are often encountered when using Gaussian processes as priors. The first is that, unless there is some special structure in the kernel matrices, densities like $p(f_D)$, $p(f_*|f_D)$ and in the Gaussian likelihood case $p(f_D, f_*|Y)$ all need to use computational operations that require $\mathcal{O}(N^3)$ flops. This is because we need the inverse and determinant of the matrices $K_{D,D}$ or Δ. This cubic scaling means that such Gaussian process methods require approximation of the exact posterior to scale to large datasets. The second challenge is the lack of a closed form for posterior marginal densities like $p(f_D, f_*|Y)$ when the likelihood is not Gaussian.

A major theme of this thesis is that both the first issue alone and the two issues simultaneously, can be addressed within the framework of variational inference. We introduce the necessary variational inference concepts in the next chapter.

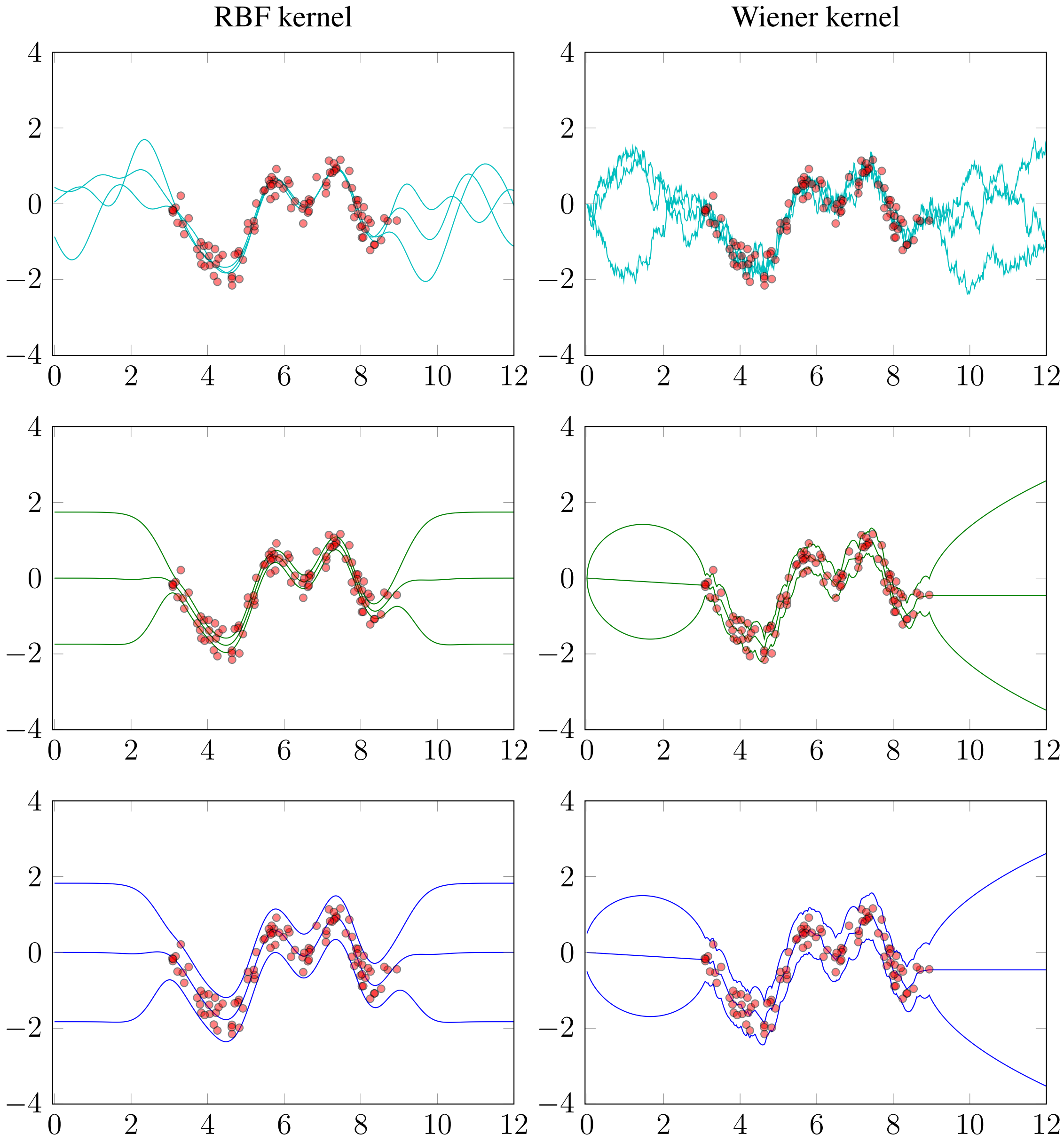

Fig. 1.3 Gaussian process regression posteriors for the Snelson dataset with a Gaussian likelihood and two different kernels. Kernel and likelihood hyperparameters have been chosen by type-II maximum likelihood (see text). Top row: 3 posterior function samples for the two different kernels. Middle row: posterior mean and 2-σ credible intervals for the posterior function. Third row: posterior mean and 2-σ credible intervals for the posterior function for the output space. Left column: posterior for the RBF kernel. The learnt hyperparameters are $\sigma = 0.87$, $l = 0.61$ and $\epsilon = 0.27$. The optimal log marginal likelihood was -33.8. Right column: posterior for the Wiener kernel. The learnt hyperparameters are $\lambda = 0.92$, $l = 1.15$ and $\epsilon = 0.25$, Note, however, that because of the rescaling symmetry described in section 1.1.3 there is a whole family of equivalent λ and l combinations. The optimal log marginal likelihood was -42.5.

Chapter 2

Variational inference

This introductory chapter emphasizes the variational inference ideas that are most relevant for the rest of this thesis. There are several good traditional introductions to variational inference available, for instance that of Jordan et al (1999).

Variational inference is a method for approximating a distribution $\hat{P}$ with a tractable distribution Q from a family of distributions $\mathcal{Q}$ using $\mathcal{KL}$-divergence as a measure of similarity. Since ultimately the distributions we wish to work with are infinite dimensional we will require a sufficiently general definition of $\mathcal{KL}$-divergence. This will be explained in the next section. After that, we will review various methods for choosing approximating families $\mathcal{Q}$. This section will mostly only require the more familiar definition of $\mathcal{KL}$-divergence given in equation (2.8). Finally we will give some simple empirical examples of the various types of variational inference.

2.1 A general definition of $\mathcal{KL}$-divergence

In this section we review the rigorous definition of the $\mathcal{KL}$-divergence between stochastic processes. We adapt the exposition of Gray (2011) who in turn credits this formulation of $\mathcal{KL}$-divergence to "Kolmogorov and his colleagues Gelfand, Yaglom, Dobrushin and Pinkser" (Kolmogorov et al., 1992). This intuitive general definition of $\mathcal{KL}$-divergence is, we argue, under utilized in literature where it is directly relevant. We therefore give a relatively brief, self-contained exposition of the points most relevant to variational inference.

The common definition of the $\mathcal{KL}$-divergence for the simpler case of two C element discrete probability distributions $u = (u_i)_{i=1}^{C}$ and $v = (v_i)_{i=1}^{C}$ is

$$\mathcal{KL}[u||v] = \sum_{i=1}^{C} u_i \log \frac{u_i}{v_i} \,. \tag{2.1}$$

We take $x \log x$, for $x = 0$ to be the corresponding limit, which is 0. The $\mathcal{KL}$-divergence may be infinite. This will occur if there is some j such that $v_j = 0$ and $u_j \neq 0$. $\mathcal{KL}$-divergence is 0 if and only if $u = v$. In all other cases $\mathcal{KL}[u||v] \geq 0$. $\mathcal{KL}$-divergence is convex in both arguments. It is not symmetric in its arguments and does not fulfil the formal mathematical requirements of a metric.

Now consider the general case where we have two measures μ and η for a regular measurable space (Ω, Σ). We are interested in dividing the space Ω into a finite measurable partition $B = \{B_i\}_{i=1}^{|B|}$. See figure 2.1 (left) for a schematic. Let $\mathcal{B}$ be the set of all such finite measurable partitions. To each such partition B there is a natural corresponding discrete distribution u from the measure μ. We evaluate the measure on the partition sets

$$u_i = \mu(B_i) \quad i = 1, \ldots, |B| \tag{2.2}$$

and similarly we can obtain a discrete distribution v using η. We can then define a discrete $\mathcal{KL}$-divergence for the partition by

$$\mathcal{KL}_B[\mu||\eta] = \sum_{i=1}^{|B|} \mu(B_i) \log \frac{\mu(B_i)}{\eta(B_i)} \,. \tag{2.3}$$

For a given partition B a *refinement* of B is any partition $\hat{B}$ that can be generated by further subdividing the elements of B. See figure 2.1 (right) for a schematic. In such a circumstance the discrete $\mathcal{KL}$-divergence has the property

$$\mathcal{KL}_{\hat{B}}[\mu||\eta] \geq \mathcal{KL}_B[\mu||\eta] \,. \tag{2.4}$$

We define the general $\mathcal{KL}$ divergence as the supremum of the corresponding discrete $\mathcal{KL}$-divergence over the set of all finite measurable partitions

$$\mathcal{KL}[\mu||\eta] := \operatorname*{Sup}_{B \in \mathcal{B}} \{\mathcal{KL}_B[\mu||\eta]\} \,. \tag{2.5}$$

Intuitively, the general $\mathcal{KL}$-divergence is defined in terms of a limiting operation on the measure space where we take the simple $\mathcal{KL}$-divergence between more and more finely grained discretizations.

A measure μ is absolutely continuous with respect to the measure η if

$$\eta(E) = 0 \implies \mu(E) = 0 \;\; \forall E \in \Sigma \tag{2.6}$$

We say that η *dominates* μ and denote this $\mu \ll \eta$. It is a theorem (Gray, 2011) that

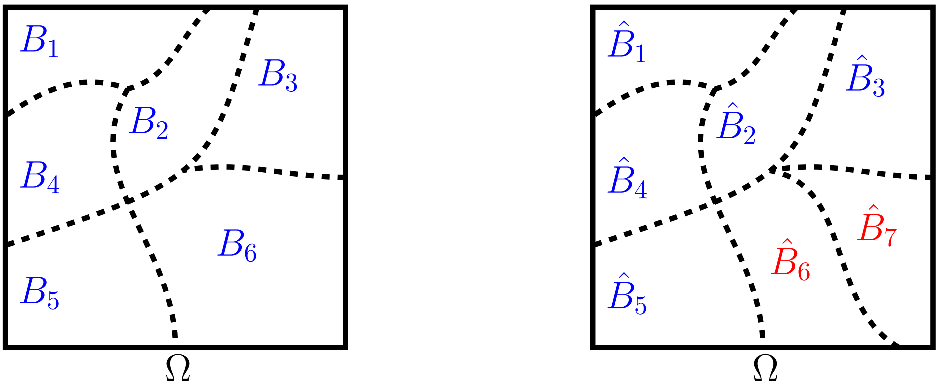

Fig. 2.1 Left: Dividing the sample space Ω into a finite measurable partition B. Right: Another partition $\hat{B}$ of Ω that is a refinement of the partition on the left.

$$\mathcal{KL}[\mu||\eta] = \begin{cases} \mathbb{E}_\mu \left\{ \log \frac{d\mu}{d\eta} \right\}, & \text{if } \mu \ll \eta \\ \infty, & \text{otherwise}\,. \end{cases} \tag{2.7}$$

The quantity $\frac{d\mu}{d\eta}$ is the Radon-Nikodym derivative which is guaranteed to exist in the case $\mu \ll \eta$ by the Radon-Nikodym theorem (Billingsley, 1995). The second undominated case in equation (2.7) can be straightforwardly verified since the fact that η does not dominate μ implies that there exists some $C \in \Sigma$ such that $\eta(C) = 0$ and $\eta(C) \neq 0$. Taking the partition $B = \{C, \Omega \backslash C\}$ the result follows. The reader is referred to (Gray, 2011) for the rigorous proof of the relation in the dominated case.

The general partition definition of $\mathcal{KL}$-divergence inherits a number of the properties from the discrete definition. Specifically $\mathcal{KL}[\mu||\eta] \geq 0$ with equality if and only if $\mu = \eta$; the $\mathcal{KL}$-divergence is jointly convex in the two measures; and it is still neither symmetric nor a metric. It is instructive to consider two special cases. In the case where the measures μ and η correspond to discrete distributions the original definition is recovered. In the case where the sample space is $\mathbb{R}^D$ for some finite D and both measures are dominated by Lebesgue measure m this reduces to the more familiar definition:

$$\mathcal{KL}[\mu||\eta] = \int_{x \in \Omega} u(x) \log \left\{ \frac{u(x)}{v(x)} \right\} dm(x) \tag{2.8}$$

where $u(x)$ and $v(x)$ are the respective densities with respect to Lebesgue measure. The partition definition is more general. As we discuss in section 3.3.1, there is no sensible Lebesgue measure for infinite dimensional spaces, meaning that the definition in equation (2.8) does not apply. The full definition in equation (2.7), by contrast, still applies.

2.2 Defining tractable approximating families $\mathcal{Q}$

The basis of variational inference is to choose a distribution Q^* to approximate an intractable distribution $\hat{P}$ using the following criterion:

$$Q^* = \arg\min_{Q \in \mathcal{Q}} \mathcal{KL}[Q||\hat{P}] \tag{2.9}$$

We will not always make a strict notational distinction between Q^* and Q -the meaning will be apparent from the context. We should also be aware that the term ‘intractable’ can take on at least three meanings in this context which are not mutually exclusive. Firstly it may mean that we do not have an analytic expression for $\hat{P}$ - often in the context of Bayesian inference this is because we do not have a closed form expression for the marginal likelihood. Another possible meaning is that it is intractable to evaluate $\hat{P}$ and related expectations in that this is provably $\mathcal{NP}$-hard in the worst case. Undirected graphical models can fall into this category. Finally, it may be that although a closed form expression is available for $\hat{P}$ and relevant expectations and it is possible to evaluate these expressions in polynomial time the exponent of the polynomial may be too high. For instance, without simplifying assumptions, exact Gaussian process regression can require $\mathcal{O}(N^3)$ computation time. If N is large then this scaling can be prohibitive.

The quality of the variational approximation will depend on there being some good approximations in the variational family $\mathcal{Q}$. If we are not careful, though, enlarging the family of distributions can mean that variational inference will no longer correspond to a tractable algorithm. How, then, does one choose a useful approximating family? In the now extensive literature on variational inference we wish to highlight and explain some key concepts that have emerged. These concepts are not necessarily mutually exclusive and in some cases are used together. For the most part we will use the simpler special case $\mathcal{KL}$ definition (2.8) to make this illustration.

For the rest of this chapter we adopt the following minimal notation. The latent variables $\mathbf{f} \in \mathbb{R}^D$ will have a prior density $p(\mathbf{f})$ and the data Y will have a likelihood $p(Y|\mathbf{f})$. The joint density $p(Y, \mathbf{f})$ will be assumed tractable to evaluation whereas the exact posterior density $p(\mathbf{f}|Y)$ and marginal likelihood $p(Y)$ will be assumed intractable to evaluate.

2.2.1 Factorization assumptions

The early use of variational methods for Bayesian inference emerged from their use in physics. The method for obtaining a simple approximating distribution was primarily to introduce factorization assumptions. The distribution over D-dimensional space is factorized into distribution over smaller groupings of the dimensions $c \in \mathcal{C}$.

$$q(\mathbf{f}) = \prod_{c \in \mathcal{C}} q(\mathbf{f}_c) \tag{2.10}$$

This early type of variational method is still the picture that many researchers have of it today. Under this approximating distribution, any covariance between two variables in different factorization blocks will be zero which may not be true of the exact posterior. Factorized methods often underestimate the variance of marginal distributions as demonstrated in section 2.4. Despite the strong assumptions of factorized approximations there are examples in the literature where, due to the difficulty of the problem, such methods perform better than commonly used alternatives. For example, it has been shown that the variational Bayesian hidden Markov model (MacKay, 1997) can empirically (Beal, 2003) give better answers than those obtained using maximum likelihood point estimates of the model parameters.

2.2.2 Working with free form distributions

Without additional simplifying assumptions beyond those of equation (2.10) the optimal solution in terms of $\mathcal{KL}$-divergence may be analysed. If we consider the approximating factor $q(\mathbf{f}_a)$ with all other distributions fixed the $\mathcal{KL}$-divergence can be written as

$$\mathcal{KL}\left[\prod_{c \in \mathcal{C}} q(\mathbf{f}_c) || p(\mathbf{f}|Y)\right] = \mathcal{KL}\left[q(\mathbf{f}_a) || \frac{\exp\left\{\mathbb{E}_{q(\mathbf{f}_{\mathcal{C}\setminus a})}[\log p(Y, \mathbf{f})]\right\}}{\int \exp\left\{\mathbb{E}_{q(\mathbf{f}_{\mathcal{C}\setminus a})}[\log p(Y, \mathbf{f})]\right\} d\mathbf{f}_a}\right] + k \tag{2.11}$$

where k is a constant that does not depend on $q(\mathbf{f}_a)$. $q(\mathbf{f}_{\mathcal{C}\setminus a})$ denotes the current approximating distribution for all variables that are not $\mathbf{f}_a$. The denominator $\int \exp\left\{\mathbb{E}_{q(\mathbf{f}_{\mathcal{C}\setminus a})}[\log p(Y, \mathbf{f})]\right\} d\mathbf{f}_a$ ensures that the $\mathcal{KL}$-divergence on the right hand side contains only properly normalized distributions. Since $\mathcal{KL}$-divergences are minimized when both sides are equal the optimal solution for $q(\mathbf{f}_a)$ is

$$q^*(\mathbf{f}_a) = \frac{\exp\left\{\mathbb{E}_{q(\mathbf{f}_{\mathcal{C}\setminus a})}[\log p(Y, \mathbf{f})]\right\}}{\int \exp\left\{\mathbb{E}_{q(\mathbf{f}_{\mathcal{C}\setminus a})}[\log p(Y, \mathbf{f})]\right\} d\mathbf{f}_a} \tag{2.12}$$

In some cases (Ghahramani and Beal, 2000) this results in a tractable approximation. The fact that we did not set out to parameterize the variational distribution in terms of some parametric family means that these approximations are sometimes referred to as ‘free form’.

The equation (2.12) can be derived using the calculus of variations with Lagrange multipliers to enforce the normalization constraints. This is why these approximations are called ‘variational’. However the derivation we have used shows this is unnecessary in this case. Often in contemporary ‘variational inference’ no calculus of variations is used whatsoever, but the original name has continued to be used.

Generally these approximations are optimized with a fixed point iteration where we alternately update the component variational distributions. This procedure is guaranteed to improve the $\mathcal{KL}$-divergence but can be slow because it corresponds to coordinate ascent. Note, however, it is often possible to parameterize the distributions implied by the updates, resubstitute into the $\mathcal{KL}$-divergence formula and use direct optimization. Since this allows joint updates it can be faster.

2.2.3 Fixed form assumptions

By contrast to free form variational inference it is possible to use ‘fixed form’ variational inference. The idea here is that the functional form for the approximating distribution is not the consequence of some free form derivation as above. Instead we simply assume that Q lies in some directly specified parametric family.

Consider a latent Gaussian model for $\mathbf{f} \in \mathbb{R}^D$ with D finite

$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K}) \tag{2.13}$$

where the likelihood is a product of terms $p(Y_i|\mathbf{f}_i)$, one for each $i = 1, ..., D$. Of particular relevance to this thesis is the case where the approximating distribution is in the family of multivariate normal distributions. We obtain the following $\mathcal{KL}$-divergence

$$\mathcal{KL}[\,q(\mathbf{f})\,||\,p(\mathbf{f}|Y)\,] = \mathcal{KL}[\,q(\mathbf{f})\,||\,p(\mathbf{f})\,] - \left[\sum_{i=1}^{D} \int q(\mathbf{f}_i) \log p(Y_i|\mathbf{f}_i) d\mathbf{f}_i\right] + \log p(Y)\,. \tag{2.14}$$

where $p(\mathbf{f})$ corresponds to the prior. As a $\mathcal{KL}$-divergence between two normal distributions the $\mathcal{KL}$-term on the right hand side can be computed in $\mathcal{O}(D^3)$. Each term in the sum on the right hand side is a one dimensional quadrature under a marginal distribution of the variational approximation. There is the question of parameterization of the covariance of the approximating distribution which could in general require $\mathcal{O}(D^2)$ parameters. However, Opper and Archambeau (2009) showed that by exploiting specific features of this problem, one can use only D covariance parameters without loss of approximation quality.

An obvious advantage of such a correlated structured approach relative to the corresponding fully factored method is that it is possible to capture posterior covariance in the approximation.

2.2.4 Prior conditional matching

The *prior conditional matching* approach to variational inference and its application to Gaussian processes will be the main topic of this thesis. It is most relevant to high dimensional problems. Instead of using that the approximating distribution factorizes as in section 2.2.1 we assume a different tractable structure. In particular partition the latent vector $\mathbf{f}$ into *inducing outputs* $\mathbf{f}_Z$ and the remainder $\mathbf{f}_{\setminus Z}$. For finite dimensional latents the assumption can then be expressed as

$$q(\mathbf{f}) = p(\mathbf{f}_{\setminus Z}|\mathbf{f}_Z)q(\mathbf{f}_Z)\,. \tag{2.15}$$

where $p(\mathbf{f}_{\setminus Z}|\mathbf{f}_Z)$ is the corresponding prior conditional distribution. That variational distribution is specified by the choice of inducing outputs $\mathbf{f}_Z$ and the distribution $q(\mathbf{f}_Z)$. For the latent Gaussian model of section 2.2.3, a later section (3.2), will show (using a different notation) that

$$\mathcal{KL}[\,q(\mathbf{f})\,||\,p(\mathbf{f}|Y)\,] = \mathcal{KL}[\,q(\mathbf{f}_Z)\,||\,p(\mathbf{f}_Z)\,] - \left[\sum_{i=1}^{D}\int q(\mathbf{f}_i)\log p(Y_i|\mathbf{f}_i)d\mathbf{f}_i\right] + \log p(Y)\,. \tag{2.16}$$

A key step in this derivation is the cancellation of the prior conditionals $p(\mathbf{f}_{\setminus Z}|\mathbf{f}_Z)$ in the ratio of the approximating and posterior densities. Hence the term *prior conditional matching*. The full infinite dimensional derivation, necessary for a general Gaussian process, is somewhat more technically involved and is given in section 3.3. The resulting expression for the $\mathcal{KL}$-divergence is very similar.

In the Gaussian process case, the $\mathcal{KL}$-divergence on the right hand side is tractable in $\mathcal{O}(M^3)$ where M is the dimensionality of $\mathbf{f}_Z$. Evaluating the sum on the right hand side requires $\mathcal{O}(NM^2)$ computation. This therefore can represent a substantial speed up over methods which require $\mathcal{O}(N^3)$. It is informative to compare equation (2.14) and equation (2.16). They are equivalent when $M = D$.

It turns out that the prior conditional matching scheme for a Gaussian process with an infinite index set is functionally equivalent to the work of Titsias (2009a), which was originally derived using an 'augmentation argument'. We offer a critique of the augmentation argument in chapter 3. We would argue that as well as putting the basic method on a firmer logical footing it is also much clearer how to correctly generalize it and we give examples of this in section 3.6.

Although they have different starting points the two different viewpoints both agree that we should use the right hand side of equation (2.16). Titsias showed that for a Gaussian process prior with a Gaussian likelihood we need make no further assumption on the nature of $q(\mathbf{f}_Z)$. In the language of this chapter an analytic free form solution $q^*(\mathbf{f}_Z)$ is available and it is a Gaussian distribution. For the non-conjugate case there is no such property and instead in chapter 4 a fixed form Gaussian distribution is *assumed*. Although in the non-conjugate case the free form solution is not available in closed form we can characterize it up to a constant. This means we can draw samples from the optimal free form distribution using MCMC. This approach, with an extension to fully Bayesian models, is investigated in chapter 5.

2.3 Treatment of hyperparameters

As discussed in section 1.2.3, the values of model hyperparameters can be difficult to choose a fixed value for a priori and can have a significant effect on the learnt posterior. We need an approach to hyperparameter selection that does not require access to the marginal likelihood $p(Y)$ which is by assumption intractable. We consider a general pattern that arises for variational inference. The $\mathcal{KL}$-divergence always take the following form

$$\mathcal{KL}[Q||\hat{P}] = -\mathcal{L} + \log p(Y)\,. \tag{2.17}$$

Here $\mathcal{L}$ will need to be tractable for variational inference to be applicable. The marginal likelihood $p(Y)$ is not a function of Q, so we can minimize $-\mathcal{L}$ instead of the $\mathcal{KL}$-divergence. This has the effect that we can only know the numerical value of the $\mathcal{KL}$-divergence up to an unknown constant. Rearranging the above equation gives

$$\mathcal{L} = -\,\mathcal{KL}[Q||\hat{P}] + \log p(Y)\,. \tag{2.18}$$

which implies, since $\mathcal{KL}$-divergences are non-negative, that $\mathcal{L}$ is a lower bound on the log marginal likelihood. Another name for the marginal likelihood is the *model evidence* so the quantity $\mathcal{L}$ is sometimes called the *evidence lower bound* or *ELBO*. If we succeed in making the $\mathcal{KL}$ term small then the value of $\mathcal{L}$ should be representative of the value of $\log p(Y)$. One way, then, to choose hyperparameters is to treat $-\mathcal{L}$ as a surrogate for $-\log p(Y)$ and minimize the hyperparamers jointly with the variational parameters to give an *approximate* empirical Bayes method.

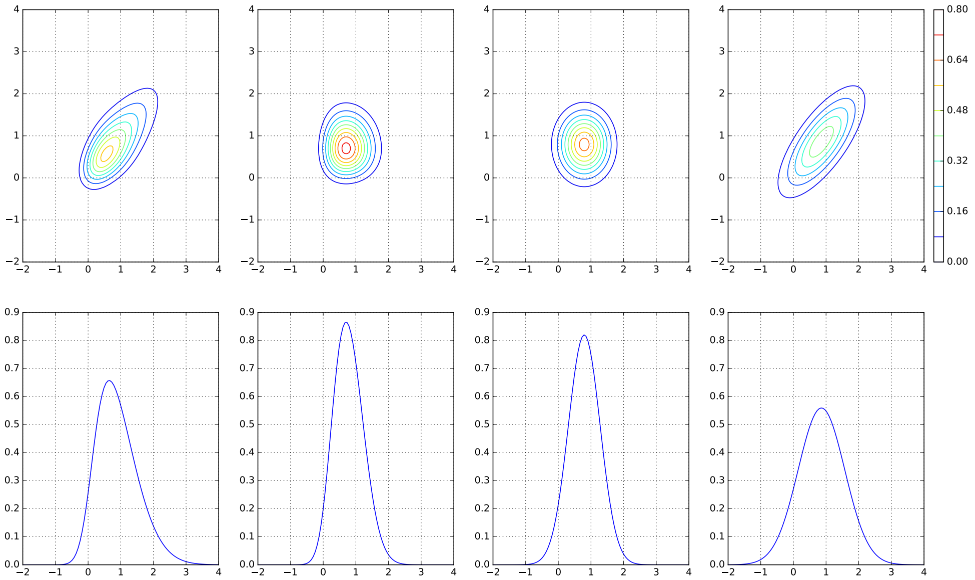

Fig. 2.2 A comparison of three different classical variational approximation methods on a two dimensional toy problem, as described in section 2.4. Top row: contours of the two dimensional density. Bottom row: a one dimensional marginal. This problem has a symmetry which makes both one dimensional marginals identical. Columns from left to right: The exact posterior which corresponds to a Gaussian latent model with a classification likelihood (see text). The factorized mean field approximation to the posterior. The factorized Gaussian approximation to the posterior. The structured Gaussian approximation to the posterior.

The slack in the bound (2.18) can be different for different values of the hyperparameters. This can bias the hyperparameter estimates towards settings with less slack (Turner and Sahani, 2011). A useful intuition is that the bias will be towards hyperparameter settings where variational inference works relatively well in terms of achieving a lower $\mathcal{KL}$-divergence. We shall see plenty of examples in this thesis where, despite this potential issue, the variational methods applied give useful answers.

As discussed in section 1.2.3, instead of a point estimate we can also use a fully Bayesian model and then make a variational approximation that includes the hyperparameter posterior. Chapter 5 adopts such an approach.

2.4 An example of classical variational inference approaches

In this section we exhibit three of the classical approaches to variational inference that we have discussed by showing the approximation to the posterior of a low dimensional toy problem. The problem in question has two latent variables which are given a Gaussian prior

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \beta \\ \beta & 1 \end{bmatrix} \right) . \tag{2.19}$$

The parameter β is taken to be 0.85. The likelihood is a product of two terms given by

$$p(Y_i|f_i) = \Phi(\gamma Y_i) \tag{2.20}$$

where Φ corresponds to the Gaussian cumulative density function and $Y \in \{-1, 1\}$. This likelihood is used as a binary classification likelihood. We set the parameter γ to be 3.0 and Y_i to be 1 for all i. The resulting posterior is symmetric between f_1 and f_2.

The optimal factorized mean field solution obtained from using equation (2.12) must have the form

$$q_i^*(f_i) = \frac{p(Y_i|f_i)\mathcal{N}(f_i|a_i, \sigma_i^2)}{Z_i} \tag{2.21}$$

for some setting of the variables a_1 and a_2. The term σ_i^2 denotes the prior variance of f_i conditioned on all the other latent Gaussian variables $f_{\setminus i}$. It is a property of multivariate Gaussian distributions that σ_i^2 is a function of the prior covariance only and is independent of the value of $f_{\setminus i}$. Z_i is a normalizing constant. The optimal values of the parameters a_i may be recovered using optimization. Since the likelihood in our example is non-Gaussian the free form mean field marginals will also be non-Gaussian. The marginals of this approximation can be viewed as ‘adapted’ to the likelihood $p(Y_i|f_i)$ since it appears as a factor in the optimal functional form of equation 2.21. We implement this approximation using a similar approach to that of Nickisch and Rasmussen (2008).

We also test the fixed form correlated Gaussian method of Opper and Archambeau 2009 which is described in section 2.2.3. Finally we also investigate the structured factorizing Gaussian approximation to this posterior. This will do strictly worse in terms of $\mathcal{KL}[Q||\hat{P}]$ then either free form mean field, or a fixed form correlated Gaussian because the variational family is a strict subset of either of these families. Nevertheless this factorized Gaussian approach is one of the most commonly used in practice due to the ease with which it may be implemented.

The results are shown in figure 2.2. There is correlation between latent variables in the posterior which both factorized approximations are unable to capture. Further, the factorized methods underestimate the marginal variance. The correlated Gaussian method does a better method of catching the correlation in the joint distribution and the variance of the marginal distribution. However, since the approximating marginal is Gaussian it cannot perfectly recover the shape of the true marginal.

2.5 An example of prior conditional matching

In this section we show a simple example of variational inference for Gaussian process regression with a Gaussian likelihood. We use the Snelson dataset with 100 training points and 100 hold out testing points. First we show variational inference with fixed hyperparameters set to the exact type II maximum likelihood values. The predictions made by the optimal variational solution are shown in figure 2.3 for a variety of inducing point numbers. The exact posterior is also shown. We see that for very small numbers of inducing points there is a clear difference between the approximate posterior and the exact posterior but that this decreases gradually until the difference is virtually imperceptible at $M = 16$. Since $N = 100$ in this case this represents a modest computational saving. We will encounter examples of much larger savings during this thesis. Note that the optimized inducing point positions are spread across the range of the data and give little coverage to the region outside. This makes sense as this is where the posterior differs most from the prior. In general the variational inducing point placement is more subtle than simply choosing the areas of highest input point density. For instance, as we shall see in sections 4.5.2 and 4.5.5, in classification problems the variational inducing point placement is often near the decision boundary.

Next, again with fixed hyperparameters, we quantify the similarity between the posterior and the approximate posterior. As metrics of similarity we use $\mathcal{KL}[Q||\hat{P}]$ which is the approximating objective and $\mathcal{KL}[\hat{P}||Q]$ which has rather different properties. Recall that in both cases the $\mathcal{KL}$-divergence is 0 if and only if the two input measures are equal. Since computing $\mathcal{KL}[\hat{P}||Q]$ involves integration under the exact posterior it requires $\mathcal{O}(N^3)$ computation and is thus only possible for small examples like this one. Similarly, although we can efficiently compute $\mathcal{KL}[Q||\hat{P}]$ up to an additive constant $\log p(Y)$, computation of the absolute value requires $\mathcal{O}(N^3)$. In both of these cases the $\mathcal{KL}$-divergence used is the full $\mathcal{KL}$-divergence between general measures introduced in section 2.1 and discussed in detail for variational inference with Gaussian processes in chapter 3. The results are shown in figure 2.4 (top and middle). We see that $\mathcal{KL}[Q||\hat{P}]$ monotonically decreases as we increase M. In fact this is provably the case as we show in section 3.3.4. By the time

$M = 12$ the approximation has negligible distortion at the scale of the plot according to this metric. $\mathcal{KL}[\hat{P}||Q]$ would generally be considered a more challenging metric for this type of variational inference. Indeed we see that, initially, adding additional inducing points causes an increase in $\mathcal{KL}[\hat{P}||Q]$. However beyond $M = 4$ this $\mathcal{KL}$-divergence also decreases and it is negligible at the scale of the plot by $M = 14$.

Finally, we show that including hyperparameter learning using the approximate empirical Bayes method described in section 2.3 still gives a good approximation for this example. Since we don't have a posterior over hyperparameters in this case there isn't a sensible $\mathcal{KL}$-divergence measure to use. Instead, we compare the quality of predictions with the exact model, in terms of predictive likelihood on hold out data. We do this for a range of inducing points numbers. As was the case for $\mathcal{KL}[\hat{P}||Q]$ the hold out negative log likelihood initially increases. As we increase the number of inducing points beyond $M = 4$ the quality improves again and once $M = 14$ the approximation gives indistinguishable prediction quality from the exact model.

This example investigated the variational approximation of a posterior Gaussian process with another Gaussian process that was chosen to be less computationally demanding to work with. The approximation of the non-Gaussian posterior processes that arise from non-Gaussian likelihoods is one of the main topics of this thesis. It is studied in chapter 4 using a Gaussian process as the approximating distribution. Complexity is added to the model by using a Bayesian hyper prior in chapter 5 where a more flexible approximating distribution is also used.

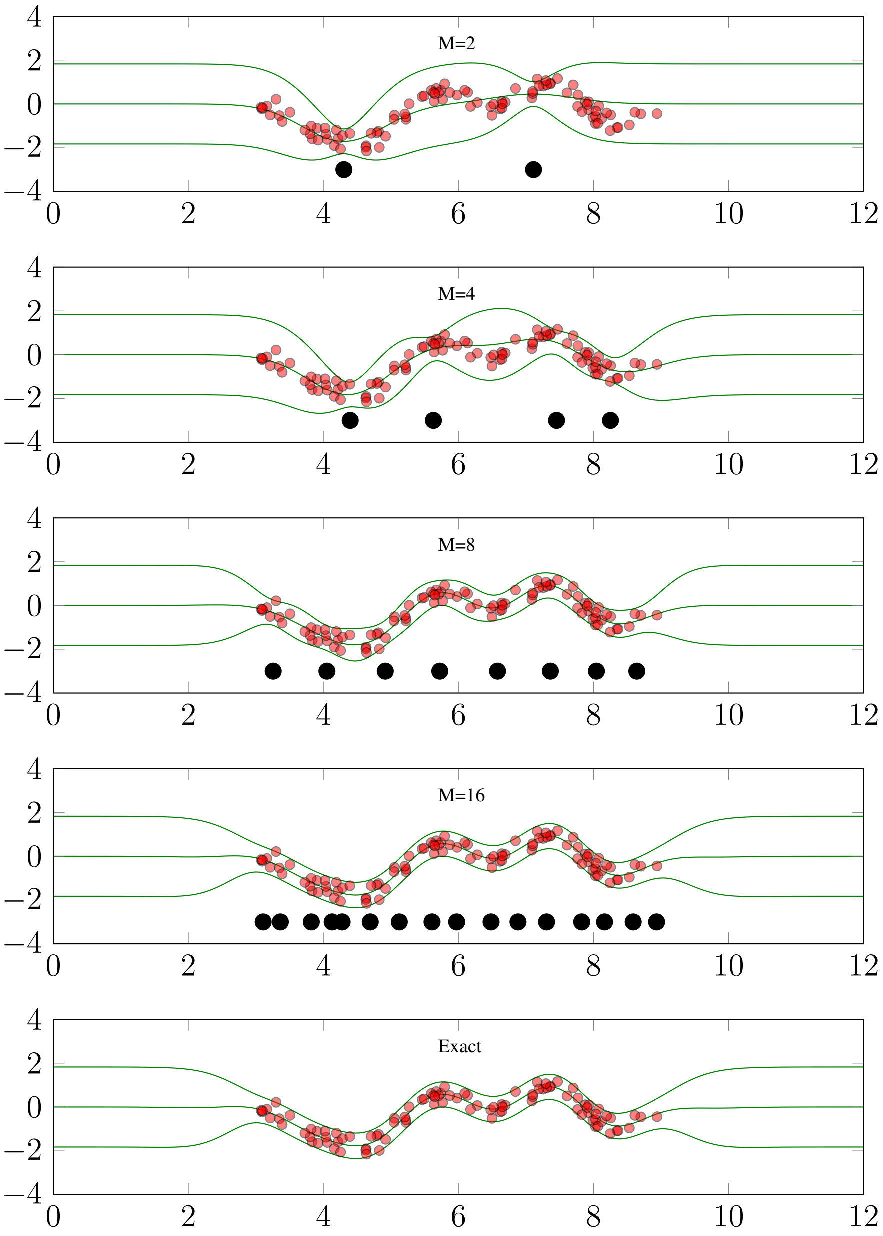

Fig. 2.3 An example of a sparse variational approximation to a Gaussian process regression posterior. The different panels show the optimal solution with a varying number of inducing points and hyperparameters fixed to the optimal type II maximum likelihood value. The data points are shown in red. The inducing input positions are shown as black markers (the output position of these markers is chosen for visualisation and has no other significance). The green lines shown the mean and 2-σ predictive credible intervals given by the approximation. The bottom plot shows the exact posterior for comparison.

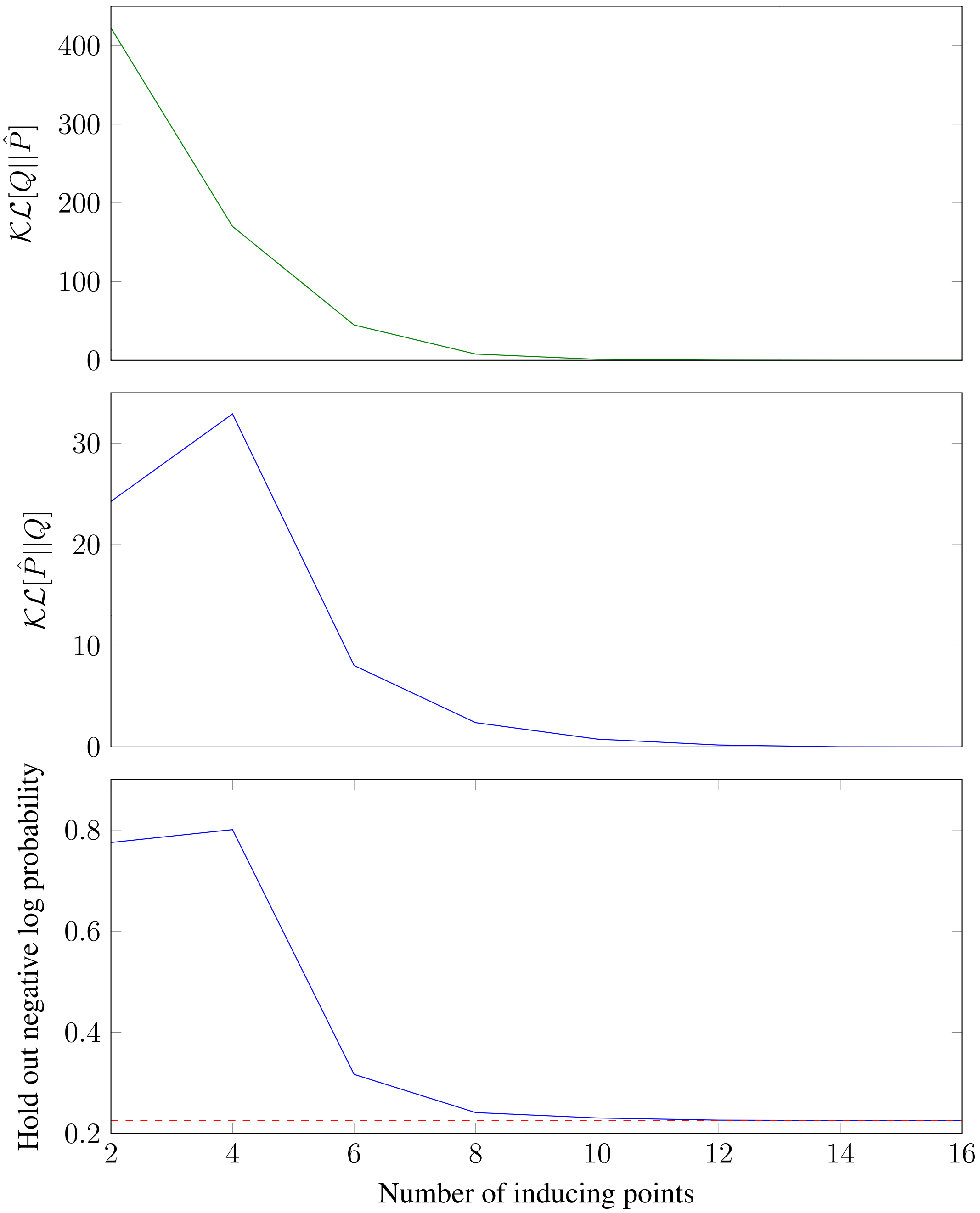

Fig. 2.4 Quantitative metrics of approximation quality for the problem shown in figure 2.3. The top and middle figures shows both possible $\mathcal{KL}$-divergences between the approximating process and the posterior process, both with hyperparameters fixed to the optimal type II maximum likelihood value. The bottom panel shows the hold out log likelihood for inducing point approximations obtained when hyperparameters are also optimized. The hold out log likelihood for predictions from the exact posterior are shown as a dashed red line.

Chapter 3

On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes

3.1 Introduction

The variational approach to inducing point selection of Titsias (2009a) has been highly influential in the active research area of scalable Gaussian process approximations. The chief advantage of this particular framework is that the inducing points positions are variational parameters rather than model parameters and as such are protected from overfitting. In this chapter we argue that whilst this is true, it may not be for exactly the reasons previously thought. The original framework is applied to conjugate likelihoods and has been extended to non-conjugate likelihoods (Chai 2012, Chapter 4). An important advance in the use of variational methods was their combination with stochastic gradient descent (Hoffman et al., 2013) and the variational inducing point framework has been combined with such methods in the conjugate (Hensman et al., 2013) and non-conjugate cases (Chapter 4). The approach has also been successfully used to perform scalable inference in more complex models such as the Gaussian process latent variable model (Damianou et al., 2015; Titsias and Lawrence, 2010) and the related Deep Gaussian process (Damianou and Lawrence, 2013; Hensman and Lawrence, 2014).

We will adopt the notation used in section 1.1.1 and section 1.2.2. For simplicity, we will initially assume that we have one, possibly noisy, possibly non-conjugate observation $y \in Y$ per input data point $d \in D$.

Gaussian processes allow us to define a prior over functions f. After we observe the data we will have some posterior which we wish to approximate with a sparse distribution. At the heart of the variational inducing point approximation is the idea of ‘augmentation’ that appears in the original paper and many subsequent ones. We choose to monitor a set $Z \subseteq X$ of size M. These points may have some overlap with the input data points D, but to give a computational speed up M will need to be fewer than the number of data points N. The Kullback-Leibler divergence given as an optimization criterion in Titsias’ original paper is

$$\begin{aligned}&\mathcal{KL}[q(f_{D\setminus Z}, f_Z)||p(f_{D\setminus Z}, f_Z|Y)] \\ &= \int q(f_{D\setminus Z}, f_Z) \log \left\{ \frac{q(f_{D\setminus Z}, f_Z)}{p(f_{D\setminus Z}, f_Z|Y)} \right\} df_{D\setminus Z} df_Z \,. \end{aligned} \tag{3.1}$$

The variational distribution at those data points which are not also inducing points is taken to have the form:

$$q(f_{D\setminus Z}, f_Z) := p(f_{D\setminus Z}|f_Z)q(f_Z) \tag{3.2}$$

where $p(f_{D\setminus Z}|f_Z)$ is the prior conditional and $q(f_Z)$ is a variational distribution on the inducing points only. Under this factorization, for a conjugate likelihood, the optimal $q(f_Z)$ has an analytic Gaussian solution (Titsias, 2009a). The non-conjugate case was then studied in subsequent work (Chai 2012, Chapter 4). In both cases the sparse approximation requires only $\mathcal{O}(NM^2)$ rather than the $\mathcal{O}(N^3)$ required by exact methods in the conjugate case, or many commonly used non-conjugate approximations that do not assume sparsity.

The augmentation argument, which is discussed in detail in section 3.4.1, is justified by arguing that the model remains marginally the same when the inducing points are added. It is therefore suggested that variational inference in the augmented model, including for the parameters of said augmentation, is equivalent to variational inference in the original model, i.e that the inducing point positions can be considered to be variational parameters and are consequently protected from overfitting. For example see Titsias’ original conference paper (Titsias, 2009a), section 3 or the longer technical report version (Titsias, 2009b), section 3.1. In many cases in the literature, sparse variational methods are derived by the application of Jensen’s inequality to the marginal likelihood. In some cases, such as Hensman et al (2015b) equations (6) and (17), the slack in the bound on the marginal likelihood is precisely the $\mathcal{KL}$-divergence (3.1). Therefore, for fixed covariance hyperparameters, maximizing such a bound is exactly equivalent to minimizing this objective and the considerations that follow all apply.

In fact, whilst we applaud the excellent prior work, we will show that variational inference in an augmented model is not equivalent to variational inference in the original model. Without this justification, the $\mathcal{KL}$-divergence in equation (3.1) could seem to be a strange optimization target. The $\mathcal{KL}$-divergence has the inducing variables on both sides, so it might seem that in optimizing the inducing point positions we are trying to hit a 'moving target'. It is desirable to rigorously formulate a 'one sided' $\mathcal{KL}$-divergence that leads to Titsias' formulation. Such a derivation could be viewed as putting these important and popular methods on a firmer foundation and is the topic of this chapter. As we shall show, this cements the framework for sparse interdomain inducing approximations and sparse variational inference in Cox processes. We wish to re-emphasize our respect for the previous work and, for the avoidance of suspense, we will find that much of the existing work carries over *mutatis mutandis*. Nevertheless, we feel that most readers at the end of the chapter will agree that a precise treatment of the topic should be of benefit going forward.

In terms of prior work for the theoretical aspect, the major other references are the early work of Seeger (2003a; 2003b). In particular Seeger identifies the $\mathcal{KL}$-divergence *between processes* (more commonly referred to as a relative entropy in those texts) as a measure of similarity and applies it to PAC-Bayes and to subset of data sparse methods. Crucially, Seeger outlines the rigorous formulation of such a $\mathcal{KL}$-divergence which is a large technical obstacle. Here we give a shorter, more general, and intuitive proof of the key theorem. We extend the stochastic process formulation to inducing points which are not necessarily selected from the data and show that this is equivalent to Titsias' formulation. In so far as we are aware this relationship has not previously been noted in the literature. The idea of using the $\mathcal{KL}$-divergence between processes is also mentioned in the early work of Csato and Opper (Csató, 2002; Csató and Opper, 2002) but the transition from finite dimensional multivariate Gaussians to infinite dimensional Gaussian processes is not covered at the level of detail discussed here. An optimization target that in intent seems to be similar to a $\mathcal{KL}$-divergence between stochastic process is briefly mentioned in the work of Alvarez (2011). The notation used suggests that the integration is with respect to an 'infinite dimensional Lebesgue measure', which as we shall see is an argument that arrives at the right answer via a mathematically flawed route. Whilst it is rigorous, the work of Pinski et al (2015a; 2015b) is tailored for the specific requirements of Bayesian inverse problems and has a distinct, non-sparse, variational family relying on the eigenfunctions of the inverse covariance operator. Chai (2012) seems to have been at least partly aware of Seeger's $\mathcal{KL}$-divergence theorems (Seeger, 2003b) but instead uses them to bound the finite joint predictive probability of a non sparse process.

This chapter proceeds by first discussing the finite dimensional version of the full argument. This requires considerably less mathematical machinery and much of the intuition can be gained from this case. We then proceed to give the full measure theoretic formulation, giving a new proof that allows inducing points that are not data points and for the likelihood to depend on infinitely many function values. These theorems enable us to show that a variety of sparse variational approximations are guaranteed to improve in terms of $\mathcal{KL}$-divergence as additional inducing points are added. Next we discuss augmentation of the original index set, using the crucial chain rule for $\mathcal{KL}$-divergences. This gives us a framework to discuss marginal consistency and how variational inference in augmented models is not necessarily equivalent to variational inference in the original model. Titsias' original argument, which will be considered in detail, is a special case of this augmentation argument. We then describe approximation in terms of *variational inducing random variables*, which allows a larger family of approximating distributions, whilst maintaining the posterior process as its optimization objective. We apply our results to sparse variational interdomain approximations and to posterior inference in Cox processes. Finally we conclude and highlight avenues for further research.

3.2 Finite index set case

This section is in fact a less general case of what follows. It is included for the benefit of those familiar with the previous work on variational sparse approximations and as an important special case. Consider the case where X is finite[1]. We introduce a new set $* := X \backslash (D \cup Z)$, in words: all points that are in the index set that are not inducing points or data points. These points might be of practical interest, for instance when making predictions on hold out data. We extend the variational distribution to include these points:

$$q(f_*, f_{D\backslash Z}, f_Z) := p(f_*, f_{D\backslash Z} | f_Z) q(f_Z). \tag{3.3}$$

We can then consider the $\mathcal{KL}$-divergence between this extended variational distribution and the full posterior distribution $p(f|Y)$

[1] We will assume in this section that all relevant distributions have density with respect to Lebesgue measure and well behaved conditional distributions.

$$
\begin{aligned}
&\mathcal{KL}[q(f)||p(f|Y)] \\
=&\mathcal{KL}[q(f_*, f_{D\backslash Z}, f_Z)||p(f_*, f_{D\backslash Z}, f_Z|Y)] \\
=&\int q(f_*, f_{D\backslash Z}, f_Z) \log \frac{q(f_*, f_{D\backslash Z}, f_Z)}{p(f_*, f_{D\backslash Z}, f_Z|Y)} df_* df_{D\backslash Z} df_Z \,. \qquad (3.4)
\end{aligned}
$$

With these definitions we are now ready to state our first theorem.

Theorem 1. *For finite index sets X, define the variational approximation as in equation* (3.3) *and the $\mathcal{KL}$-divergence as in equation* (3.4)*. Then the following two properties hold*

(i) The $\mathcal{KL}$-divergence across the whole index set X may be decomposed in the following way

$$\mathcal{KL}[q(f)||p(f|Y)] = \mathcal{KL}[q(f_Z)||p(f_Z)] - \mathbb{E}_{Q_D}[\log p(Y|f_D)] + \log p(Y)$$

where Q_D is the marginal distribution of the variational approximation at the data inputs D.

(ii) The $\mathcal{KL}$-divergence across the whole index set X is equal to Titsias' $\mathcal{KL}$-divergence described in equation (3.1)

$$\mathcal{KL}[q(f)||p(f|Y)] = \mathcal{KL}[q(f_{D\backslash Z}, f_Z)||p(f_{D\backslash Z}, f_Z|Y)] \,.$$

Proof. To prove part (i) of the theorem we start from equation (3.4). Substituting the definition for the approximating density given in equation (3.3) and the definition of the posterior using Bayes' theorem and the chain rule we obtain

$$
\begin{aligned}
&\mathcal{KL}[q(f)||p(f|Y)] \\
=&\int p(f_*, f_{D\backslash Z}|f_Z) q(f_Z) \log \left\{ \frac{p(f_*|f_{D\backslash Z}, f_Z) p(f_{D\backslash Z}|f_Z) q(f_Z) p(Y)}{p(f_*|f_{D\backslash Z}, f_Z) p(f_{D\backslash Z}|f_Z) p(f_Z) p(Y|f_D)} \right\} df_* df_{D\backslash Z} df_Z \\
=&\int p(f_*, f_{D\backslash Z}|f_Z) q(f_Z) \log \left\{ \frac{q(f_Z) p(Y)}{p(f_Z) p(Y|f_D)} \right\} df_* df_{D\backslash Z} df_Z \,. \qquad (3.5)
\end{aligned}
$$

Note how the choice of approximating distribution has enabled us to cancel some terms inside the logarithm. Specifically the prior conditional densities $p(f_*|f_{D\backslash Z}, f_Z)$ and $p(f_{D\backslash Z}|f_Z)$ have cancelled. It is for this reason that we call such an approximation assumption *prior conditional matching* (see section 2.2.4). Next we split the integral into a sum of three terms using the multiplication property of the logarithm and then apply the marginalization property to $p(f_*, f_{D\setminus Z}|f_Z)$

$$\begin{aligned}&\mathcal{KL}[q(f)||p(f|Y)]\\ &= \int q(f_Z)\log\left\{\frac{q(f_Z)}{p(f_Z)}\right\}df_Z - \int p(f_{D\setminus Z}|f_Z)q(f_Z)\log p(Y|f_D)df_{D\setminus Z}df_Z + \log p(Y)\,.\end{aligned} \tag{3.6}$$

Identifying the first two terms as $\mathcal{KL}[q(f_Z)||p(f_Z)]$ and $-\mathbb{E}_{Q_D}[\log p(Y|f_D)]$ respectively, part (i) of the theorem is proved.

To show part (ii) of the theorem it is sufficient to show that

$$\mathcal{KL}[q(f_{D\setminus Z}, f_Z)||p(f_{D\setminus Z}, f_Z|Y)] = \mathcal{KL}[q(f_Z)||p(f_Z)] - \mathbb{E}_{Q_D}[\log p(Y|f_D)] + \log p(Y) \tag{3.7}$$

since this can be combined with (i) to give the desired result. The equality in equation 3.7 follows from substituting the necessary densities. □

Theorem 1.ii shows us that with this interpretation, the appearance of the inducing values on both sides of Titsias' $\mathcal{KL}$-divergence in equation (3.1) is just a question of 'accounting'. That is to say, whilst we are in fact optimizing the $\mathcal{KL}$-divergence between the full distributions, we only need to keep track of the distribution over function values f_Z and $f_{D\setminus Z}$. All the other function values f_* marginalize. We might choose to optimize our choice of the M inducing inputs Z by selecting them from the $|X|$ points in the index set. For different choices of inducing points we will need to keep track of different function values and be able to safely ignore different values f_*.

It will be useful in what follows to note that we can use our notation to write

$$\mathcal{KL}[q(f_Z)||p(f_Z)] = \mathcal{KL}[Q_Z||P_Z] \tag{3.8}$$

and

$$\mathcal{KL}[q(f_{D\setminus Z}, f_Z)||p(f_{D\setminus Z}, f_Z|Y)] = \mathcal{KL}[Q_{Z\cup D}||\hat{P}_{Z\cup D}]\,. \tag{3.9}$$

The benefit is that when we relax the assumption of this section that densities like $q(f_Z)$ exist, we will still be able to use the more general notation on the right hand side of equations (3.8) and (3.9).

3.3 Infinite index set case

3.3.1 There is no useful infinite dimensional Lebesgue measure

One might hope to cope with not only finite index sets but also infinite index sets in the way discussed in section 3.2. Unfortunately when X and hence f_* are infinite sets we cannot integrate with respect to a 'infinite dimensional vector'. That is to say the notation $\int(\cdot)df_*$ can no longer be correctly used.

For a discussion of this see, for example, the introduction of Hunt et al (1992). We briefly summarize the essentials of the argument here. In finite dimensions, one of the key properties that makes Lebesgue measure convenient to work with is that it is translation invariant. The translation invariance in turn implies a scaling property of K dimensional Lebesgue measure m. Suppose, for instance, in two dimensions that we have a square with sides of length ϵ and area a. A square with sides of length 2ϵ must necessarily have area $4a$ because, by translation invariance, we can tile the larger square with four copies of the smaller one. For general dimension K, by a similar argument, doubling the side of a hypercube of hypervolume a will give a hypercube of hyper volume $2^K a$. We can now see the issue with infinite dimensional Lebesgue measure. If a is non-zero the scaling relationship cannot hold. But a would need to be non-zero for a usable dominating measure.

Thus we see that it will be necessary to rethink our approach to a $\mathcal{KL}$-divergence between stochastic processes. It will turn out that a reasonable definition will require the apparatus of measure theory. Readers looking for some background on these issues may wish to consult a larger text (Billingsley, 1995; Capinski and Kopp, 2004).

3.3.2 The $\mathcal{KL}$-divergence between processes

We need to recall the rigorous definition of the $\mathcal{KL}$-divergence between stochastic processes which is discussed in section 2.1 and in the text of Gray (2011).

Suppose we have two measures μ and η for (Ω, Σ) and that μ is absolutely continuous with respect to η. Then there exists a Radon-Nikodym derivative $\frac{d\mu}{d\eta}$ and the correct definition for $\mathcal{KL}$-divergence between these measures is:

$$\mathcal{KL}[\mu||\eta] = \int_\Omega \log\left\{\frac{d\mu}{d\eta}\right\} d\mu\,. \tag{3.10}$$

In the case where μ is not absolutely continuous with respect to η we let $\mathcal{KL}[\mu||\eta] = \infty$. In the case where the sample space is $\mathbb{R}^K$ for some finite K and both measures are dominated by Lebesgue measure m this reduces to the more familiar definition:

$$\mathcal{KL}[\mu||\eta] = \int_{\Omega} u \log \left\{ \frac{u}{v} \right\} dm \tag{3.11}$$

where u and v are the respective densities with respect to Lebesgue measure. The first definition (3.10) is more general and allows us to deal with the problem of there being no sensible infinite dimensional Lebesgue measure by instead integrating with respect to the measure μ.

3.3.3 A general derivation of the sparse inducing point framework

In this section we derive the key equations of the sparse inducing point framework. The derivation is more general than that of Seeger (2003a; 2003b) since it does not require that the inducing points are selected from the data points. Nor does it assume that the relevant finite dimensional marginal distributions have density with respect to Lebesgue measure. Finally, since the dependence on the elegant properties of Radon-Nikodym derivatives has been made more explicit, we believe it is clearer *why* the derivation works and how one would generalize it, which will be useful in extensions.

We are now interested in three types of probability measure on sets of functions $f : X \mapsto \mathbb{R}$. The first is the prior measure P which will be assumed to be a Gaussian process. The second is the approximating measure Q which will be assumed to be a sparse Gaussian process and the third is the posterior process $\hat{P}$ which may be Gaussian or non-Gaussian depending on whether we have a conjugate likelihood. We start with a measure theoretic version of Bayes' theorem for a discriminative, dominated model, which is discussed in section 1.2.2 and by Schervish (1995). It specifies the Radon-Nikodym derivative of the posterior with respect to the prior

$$\frac{d\hat{P}}{dP}(f) = \frac{p_X(Y|f)}{p(Y)}, \tag{3.12}$$

with $p_X(Y|f)$ being the likelihood and $p(Y) = \int_{\mathbb{R}^X} p_X(Y|f) dP(f)$ the marginal likelihood. As we have assumed in previous sections, we will initially restrict the likelihood to only depend on the finite data subset of the index set. As in section 1.1.1, we denote by $\pi_C : \mathbb{R}^X \mapsto \mathbb{R}^C$ a projection function, which takes the whole function as an argument and returns the function at some set of points C. In this case we have:

$$\frac{d\hat{P}}{dP}(f) = \frac{d\hat{P}_D}{dP_D}(\pi_D(f)) = \frac{p(Y|\pi_D(f))}{p(Y)} \tag{3.13}$$

and similarly the marginal likelihood only depends on the function values on the data set $p(Y) = \int_{\mathbb{R}^D} p(Y|f_D) dP_D(f_D)$. In fact, we will relax the assumption that the data set is finite in section 3.6.2 and the ability to do so is one of the benefits of this framework.

Next we specify the approximating distribution Q by assuming it has density with respect to the posterior and thus the prior. We assume that the density with respect to the prior depends on some set of points Z:

$$\frac{dQ}{dP}(f) = \gamma(\pi_Z(f)) \,. \tag{3.14}$$

Here γ is some measurable function $\gamma : \mathbb{R}^M \mapsto [0, \infty)$. Note that the inducing output projection function π_Z like the data projection function π_D is a measurable function when using the product σ-algebra for f, because the pre-images of the Borel sets are cylinder sets (see section 1.1.1). The requirement that the marginals of the distribution Q must correctly normalize constrains the function γ. In fact, we can identify it as the Radon-Nikodym derivative $\frac{dQ_Z}{dP_Z}$ of the marginal distribution for the inducing outputs Q_Z with respect to the corresponding marginal distribution in the prior P_Z. Thus the density of the approximating distribution with respect to the prior takes the form

$$\frac{dQ}{dP}(f) = \frac{dQ_Z}{dP_Z}(\pi_Z(f)) \,. \tag{3.15}$$

Under this assumption Q is fully specified if we know P, Z and $\frac{dQ_Z}{dP_Z}$. To gain some intuition for this assumption, we can compare equations (3.15) and (3.13). We see that in the approximating distribution the set Z is playing a similar one to that played for D in the true posterior distribution. Next we give a theorem which is the generalization of theorem 1 to infinite index sets and does not require Lebesgue densities.

Theorem 2. *Let $\hat{P}$ be the posterior distribution that arises from a dominated discriminative model, as in equation* (3.13). *Let Q be the approximating distribution defined by* (3.15). *Define $\mathcal{KL}$-divergence as in equation* (3.10). *Then:*

(i) The $\mathcal{KL}$-divergence between the approximating process and the posterior process $\mathcal{KL}[Q||\hat{P}]$ obeys the equality

$$\mathcal{KL}[Q||\hat{P}] = \mathcal{KL}[Q_Z||P_Z] - \mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right] + \log p(Y) \,.$$

(ii) The $\mathcal{KL}$-divergence between processes and Titsias' $\mathcal{KL}$-divergence are equal:

$$\mathcal{KL}[Q||\hat{P}] = \mathcal{KL}[Q_{Z\cup D}||\hat{P}_{Z\cup D}] .$$

Proof. To prove part (i) we start by applying the chain rule for Radon-Nikodym derivatives and a standard property of logarithms (see for example Gray (2011), corollary 7.1):

$$\mathcal{KL}[Q||\hat{P}] = \int_{\mathbb{R}^X} \log\left\{\frac{dQ}{dP}(f)\right\} dQ(f) - \int_{\mathbb{R}^X} \log\left\{\frac{d\hat{P}}{dP}(f)\right\} dQ(f) . \tag{3.16}$$

Taking the first term alone, we exploit the sparsity assumption (3.15) for the approximating distribution and the usual marginalization property of integration:

$$\begin{aligned} &\int_{\mathbb{R}^X} \log\left\{\frac{dQ}{dP}(f)\right\} dQ(f) \\ &= \int_{\mathbb{R}^Z} \log\left\{\frac{dQ_Z}{dP_Z}(f_Z)\right\} dQ_Z(f_Z) \\ &= \mathcal{KL}[Q_Z||P_Z] \end{aligned} \tag{3.17}$$

Taking the second term in the last line of equation (3.16) then exploiting the measure theoretic Bayes' theorem (3.13) and marginalization we obtain:

$$\begin{aligned} &\int_{\mathbb{R}^X} \log\left\{\frac{d\hat{P}}{dP}(f)\right\} dQ(f) \\ &= \int_{\mathbb{R}^D} \log\left\{\frac{d\hat{P}_D}{dP_D}(f_D)\right\} dQ_D(f_D) \\ &= \mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right] - \log p(Y) . \end{aligned} \tag{3.18}$$

Finally combining equations (3.17) and (3.18) we prove this part of the theorem.

To prove part (ii) of the theorem, if the relevant densities exist we can again use equation (3.7), which in the more general notation is written as

$$\mathcal{KL}[Q_{Z\cup D}||\hat{P}_{Z\cup D}] = \mathcal{KL}[Q_Z||P_Z] - \mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right] + \log p(Y) . \tag{3.19}$$

Combining this equality with part (i) we may consequently recover the result in part (ii). To remove the dependence of equation (3.19) on densities with respect to Lebesgue measure we can note that:

$$\frac{dQ_{Z\cup D}}{dP_{Z\cup D}}(f_{Z\cup D}) = \frac{dQ_Z}{dP_Z}(\pi_{Z\cup D\to Z}(f_{Z\cup D})) \tag{3.20}$$

and that

$$\frac{d\hat{P}_{Z\cup D}}{dP_{Z\cup D}}(f_{Z\cup D}) = \frac{p(Y|\pi_{Z\cup D\to D}(f_{Z\cup D}))}{p(Y)} \,. \tag{3.21}$$

Substituting these definitions into $\mathcal{KL}[Q_{Z\cup D}||\hat{P}_{Z\cup D}]$ and following a similar argument to the proof of part (i) we obtain the result (3.19) and hence part (ii) is proven in the general case. □

Theorem 2.i is used throughout the rest of this thesis. As is common with variational approximations, in most cases of interest the marginal likelihood will be intractable. However since it is an additive constant, independent of Q, it can be safely ignored. The final equation shows that we need to be able to compute the $\mathcal{KL}$-divergence between the inducing point marginals of the approximating distribution and the prior for all $Z \subset X$ and the expectation under the data marginal distribution of Q of the log likelihood. In the case where the likelihood factorizes across data terms this will give a sum of one dimensional expectations. Note the similarity of Theorem 2.i with Hensman et al. (2015b) equation (17) where a less general expression is motivated from a 'model augmentation' view.

Theorem 2.ii is important because the right hand side of the equation is used in Titsias' paper (2009a). This means that derivations in Titsias' paper that use this $\mathcal{KL}$-divergence as a starting point also apply to the left hand side of the equation- namely the $\mathcal{KL}$-divergence between the approximating and posterior process. In other words, Titsias' approximation framework may be interpreted as minimizing a rigorously defined $\mathcal{KL}$-divergence between the approximating and posterior processes. This provides an alternative motivation to the augmentation argument that Titsias used to justify the use of $\mathcal{KL}[Q_{Z\cup D}||\hat{P}_{Z\cup D}]$ in his original paper. In section 3.4 we will discuss some problems with these augmentation style arguments.

Notice that at no point in our derivation did we try to envoke the pathological 'infinite dimensional Lebesgue measure' which is important for the reasons discussed in section 3.3.1. The ease of derivation suggests that Radon-Nikodym derivatives and measure theory provide the most natural and general way to think about such approximations.

3.3.4 The effect of increasing the number of inducing points

We now discuss circumstances under which increasing the number of inducing points will give an approximation to the posterior which has a monotonically decreasing $\mathcal{KL}$-divergence between the approximating process and the posterior process. Monotonicity is a desirable property, because loosely it tells us that expending extra computational resources in the form of extra inducing points can only improve or maintain the quality of approximation. Titsias (2009a, section 3.1) discusses the related case of greedy selection of inducing points from the training set for regression. Recently and independently, Bauer et at (2016) provide a proof for regression with general inducing points. The proof we give is more general and does not required detailed linear algebra considerations. We would also argue that it is clearer in our proof *why* the inequality must hold. Thus this section can be seen as leveraging the theory of the current work to characterize and understand a broad set of circumstances under which monotonic improvement can be guaranteed.

To prove our monotonicity result, we will need to set up some additional notation. Let $\mathcal{Q}^{(M)}$ be the set of possible M inducing point variational approximating distributions. Let $\mathcal{KL}^{(M)}_{\text{opt}}$ be the optimal $\mathcal{KL}$-divergence attainable within the variational family $\mathcal{Q}^{(M)}$, that is to say:

$$\mathcal{KL}^{(M)}_{\text{opt}} := \min_{Q^{(M)} \in \mathcal{Q}^{(M)}} \mathcal{KL}[Q^{(M)} || \hat{P}] \tag{3.22}$$

Having a rigorously defined, one sided optimization objective allows us to use the following lemma:

Lemma 1. *A sequence of variational approximations that has the property*

$$\mathcal{Q}^{(M+1)} \supseteq \mathcal{Q}^{(M)}$$

will have optimal $\mathcal{KL}$-divergences that are a monotonically decreasing function of M

$$\mathcal{KL}^{(M+1)}_{opt} \leq \mathcal{KL}^{(M)}_{opt} \ .$$

Proof. The strictly larger variational family must always give at least as small an optimal $\mathcal{KL}$-divergence, since $\mathcal{Q}^{(M+1)}$ must necessarily contain the optimal distribution from $\mathcal{Q}^{(M)}$. □

The condition of the lemma is in fact stronger than strictly necessary, since in fact we only require that $\mathcal{Q}^{(M+1)}$ contain the optimal member of $\mathcal{Q}^{(M)}$. The stronger condition is the one applied in what follows.

We have reduced the question of when the bound holds to a question about the sequence of variational approximating families as we increase the number of inducing points. Recall that the variational approximation is fully specified by two objects. Firstly, we require the positions of the inducing points $Z \in X^M$. Second, we require a probability measure on the functions at the inducing point values $Q^{(M)}_Z \in \mathcal{Q}^{(M)}_{\text{ind}}$ where $\mathcal{Q}^{(M)}_{\text{ind}}$ is the corresponding variational family for M inducing points. The next theorem gives some conditions under which such a sequence of variational distributions will have optimal $\mathcal{KL}$-divergences that are a monotonically decreasing function of M.

Theorem 3. *Suppose that for all numbers of inducing points M, sets of M inducing inputs $Z \in X^M$ and corresponding inducing output distributions $Q_Z^{(M)} \in \mathcal{Q}_{ind}^{(M)}$, there is some additional inducing input $\beta \in X$ and $M+1$ inducing output distribution $Q_{\beta \cup Z}^{(M+1)} \in \mathcal{Q}_{ind}^{(M+1)}$, such that the following condition holds:*

$$Q_{Z \cup \beta}^{(M+1)} = Q_{Z \cup \beta}^{(M)} .$$

That is to say, the $M+1$ inducing output distribution is equal to the corresponding marginal of $Q^{(M)}$. Then the optimal $\mathcal{KL}$-divergence will be a monotonically decreasing function of M

$$\mathcal{KL}_{opt}^{(M+1)} \leq \mathcal{KL}_{opt}^{(M)} .$$

We remark that in the case where the relevant densities with respect to Lebesgue measure exist, then the two conditions of the theorem correspond to there being some $\beta \in X$ such that the inducing output families obey the relation

$$q^{(M+1)}(f_\beta, f_Z) = p(f_\beta | f_Z) q^{(M)}(f_Z) . \tag{3.23}$$

Proof. The proof relies on showing that the conditions of lemma 1 are met. This will be the case if for every member of $\mathcal{Q}^{(M)}$ there corresponds at least one element $\mathcal{Q}^{(M+1)}$ which gives an equal variational distribution. The conditions of theorem 3 ensure precisely such a correspondence. Consider the definitions of the two distributions, specifically

$$\frac{dQ^{(M)}}{dP}(f) = \frac{dQ_Z^{(M)}}{dP_Z}(\pi_Z(f)) \tag{3.24}$$

and

$$\frac{dQ^{(M+1)}}{dP}(f) = \frac{dQ_{\beta \cup Z}^{(M+1)}}{dP_{\beta \cup Z}}(\pi_{\beta \cup Z}(f)) . \tag{3.25}$$

Clearly the measures $Q^{(M)}$ and $Q^{(M+1)}$ will be equal if the following relation holds

$$\frac{dQ_Z^{(M)}}{dP_Z}(\pi_Z(f)) = \frac{dQ_{\beta \cup Z}^{(M+1)}}{dP_{\beta \cup Z}}(\pi_{\beta \cup Z}(f)) . \tag{3.26}$$

Applying the condition of the theorem to the right hand side of this equation we obtain

$$\frac{dQ^{(M+1)}_{\beta \cup Z}}{dP_{\beta \cup Z}}(\pi_{\beta \cup Z}(f)) = \frac{dQ^{(M)}_{\beta \cup Z}}{dP_{\beta \cup Z}}(\pi_{\beta \cup Z}(f)) \,. \tag{3.27}$$

But for index set elements such as β which are not inducing inputs Z, $Q^{(M)}$ has the same conditionals as the prior so

$$\frac{dQ^{(M)}_{\beta \cup Z}}{dP_{\beta \cup Z}}(\pi_{\beta \cup Z}(f)) = \frac{dQ^{(M)}_{Z}}{dP_{Z}}(\pi_{Z}(f)) \,. \tag{3.28}$$

The relation (3.26) therefore holds and consequently the theorem is proved.

In the case where the relevant densities exist, the relation (3.23) would allow us to prove this result without using as much formal measure theory. □

This theorem has a number of implications for the literature which we list as corollaries.

Corollary 3.1. *For fixed hyperparameters, the uncollapsed Gaussian likelihood regression approximation of Hensman et al (2013) and the earlier collapsed Gaussian likelihood regression approximation of Titsias (2009a) have a monotonically decreasing optimal $\mathcal{KL}$-divergence as we add inducing points.*

Proof. As we have shown in this chapter, both of these frameworks can be viewed as minimizing the $\mathcal{KL}$-divergence between the approximating process Q and the posterior process $\hat{P}$. Hensman et al (2013) use an explicitly parameterized family of multivariate normal distributions for $\mathcal{Q}^{(M)}_{\text{ind}}$, with a full covariance matrix. These variational families are expressive enough to meet the conditions of the theorem. For the Titsias framework the variational family for the inducing outputs is free form. Since the free form solution is necessarily Gaussian, we can apply the same logic as we do for the method of Hensman et at (2013). The only difference for our current purposes is that the optimization of the inducing output distribution is done analytically. □

As already discussed, the special case of corollary 3.1 where the inducing inputs are selected from the data inputs was proved by Titsias (2009b) and the general result for Titsias' framework was recently proved independently using linear algebra by Bauer et al (2016).

The next two corollaries are for future reference, since they apply to the contributions of the rest of this thesis.

Corollary 3.2. *For fixed hyperparameters the non-Gaussian likelihood approximation of chapter 4 has a monotonically decreasing optimal $\mathcal{KL}$-divergence as we add inducing points.*

Proof. Since the inducing output distributions are again full covariance Gaussian, this follows from similar considerations to those applied to the work of Hensman et al (2013) in the proof of corollary 3.1. □

Corollary 3.3. *The non-Gaussian likelihood approximation of chapter 5 has a monotonically decreasing optimal $\mathcal{KL}$-divergence as we add inducing points.*

Proof. Since the hyperparameters are given a Bayesian prior and then approximated variationally in this framework, they do not need to be fixed to apply the theorem. It is necessary to modify the conditions of the theorem to allow for a joint variational distribution of inducing outputs and hyperparameters but this goes through straightforwardly. □

It is worth noting (as we discuss in section 4.2.2) that many other sparse approximation methods currently in use do not have guarantees on the approximation quality as the number of inducing points is increased. We also note that if there exists some number of inducing points M (usually this is when $M = N$) such that the true posterior distribution is a member of $\mathcal{Q}^{(M)}$ then the $\mathcal{KL}$-divergence will be zero and adding further inducing points will have no utility. An empirical study of the optimal $\mathcal{KL}$-divergence as a function of the number of inducing points M was made for a simple regression example in section 2.5.

3.4 Augmented index sets

3.4.1 The augmentation argument.

The most important example of what we term the 'augmentation argument' is illustrated by this quote from Titsias' original paper:

> *A principled procedure to specify $q(f_Z)$ and the inducing inputs Z is to form the variational distribution $q(f_D)$ and the exact posterior $p(f_D|y)$ on the training function values f_D, and then minimize a distance between these two distributions. Equivalently, we can minimize a distance between the augmented true posterior $p(f_{D\backslash Z}, f_Z|y)$ and the augmented variational posterior $p(f_{D\backslash Z}, f_Z|y)$ where clearly from eq. (5) $q(f_{D\backslash Z}, f_Z) = p(f_{D\backslash Z}|f_Z)q(f_Z)$*

— *Michalis K. Titsias*. AISTATS 2009.

The only changes we have made to this quote are to adapt the notation to our own and to allow for the possibility that there may be some intersection between the set of inducing inputs Z and the set of data inputs D . The latter change is not essential to our argument.

The quote is not accompanied by an equation, so it has a degree of ambiguity. The range of possibilities seems well covered by the following two propositions.

Proposition 1. *Using the sparse variational framework, the following equality between $\mathcal{KL}$-divergences always holds*

$$\mathcal{KL}\left[Q_D^{(\theta)}||\hat{P}_D\right] = \mathcal{KL}\left[Q_{D\cup Z}^{(\theta)}||\hat{P}_{D\cup Z}^{(\theta)}\right]$$

Proposition 2. *Using the sparse variational framework, the following equality between optima of $\mathcal{KL}$-divergences always holds*

$$\arg\min_{\theta}\left\{\mathcal{KL}\left[Q_D^{(\theta)}||\hat{P}_D\right]\right\} = \arg\min_{\theta}\left\{\mathcal{KL}\left[Q_{D\cup Z}^{(\theta)}||\hat{P}_{D\cup Z}^{(\theta)}\right]\right\}$$

We will show conclusively that neither of these propositions is true. Note that for this section only, we have explicitly shown the dependence of the distributions on those parameters θ which are treated as variational parameters. As already noted, such parameters appear on the right hand side of Titsias' $\mathcal{KL}$-divergence $\mathcal{KL}\left[Q_{D\cup Z}^{(\theta)}||\hat{P}_{D\cup Z}^{(\theta)}\right]$. Before addressing the falsehood of the propositions, we will formulate a slightly more general version of the augmentation argument, of which Titsias' argument will be a special case.

The model we have been working with up until this point is:

$$f_X \sim P_X \tag{3.29}$$

$$Y|f_D \sim p(Y|f_D)\,. \tag{3.30}$$

The idea of sparse variational augmentation is to consider some extra function values f_I generated from a distribution that my depend on some parameters θ that are alleged to be variational parameters

$$f_I|f_X,\theta \sim P_{I|X}^{(\theta)}\,. \tag{3.31}$$

Whatever the conditional distribution $P_{I|X}^{(\theta)}$ is, the augmented model will be consistent with the original model under marginalization of f_I. The approximating measure $Q_{X\cup I}$ is defined by specifying its density with respect to the augmented prior $P_{X\cup I}$. This density takes the form of a projection onto the augmented function values

$$\frac{dQ^{(\theta)}_{X\cup I}}{dP^{(\theta)}_{X\cup I}}(f_{X\cup I}) = \frac{dQ^{(\theta)}_{I}}{dP^{(\theta)}_{I}}(\pi_I(f_{X\cup I}))\,. \tag{3.32}$$

This corresponds to the original augmentation argument given by Titsias (2009a) when $X = D$ and $I = Z$. It will be seen that this is also very much in the spirit of the 'variational compression' framework of Hensman and Lawrence (2014).

Acting as if the augmented set were the original index set we obtain by a similar argument to section 3.3.3:

$$\mathcal{KL}[Q^{(\theta)}_{X\cup I}||\hat{P}^{(\theta)}_{X\cup I}] = \mathcal{KL}[Q^{(\theta)}_{I}||P^{(\theta)}_{I}] - \mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right] + \log p(Y)\,. \tag{3.33}$$

Readers will note however the appearance of the parameters θ on both sides of the left hand $\mathcal{KL}$-divergence. As with the Titsias argument, which is a special case, the question arises as to whether this is equivalent to minimizing the $\mathcal{KL}$-divergence on the original X space. To resolve this question, we will use the chain rule for $\mathcal{KL}$-divergences which we now pause to describe.

3.4.2 The chain rule for $\mathcal{KL}$-divergences

Let U and V be two Polish spaces endowed with their standard Borel σ-algebras and let $U\times V$ be the Cartesian product of these spaces endowed with the corresponding product σ-algebra. Consider two probability measures $\mu_{U\times V}, \eta_{U\times V}$ on this product space and let $\mu_{U|V}, \eta_{U|V}$ be the corresponding regular conditional measures. Assume that $\mu_{U\times V}$ is dominated by $\eta_{U\times V}$. The chain rule for $\mathcal{KL}$-divergences says that:

$$\mathcal{KL}[\mu_{U\times V}||\eta_{U\times V}] = \mathbb{E}_{\mu_V}\big\{\mathcal{KL}[\mu_{U|V}||\eta_{U|V}]\big\} + \mathcal{KL}[\mu_V||\eta_V]. \tag{3.34}$$

The first term on the right hand side is referred to as the 'conditional relative entropy'. Further detail on the chain rule for $\mathcal{KL}$-divergences can be found in the text of Gray (2011).

3.4.3 The augmentation argument is not correct in general

We are now ready to state the key result of this section as a theorem

Theorem 4. *Minimizing the augmented $\mathcal{KL}$-divergence $\mathcal{KL}[Q^{(\theta)}_{X\cup I}||\hat{P}^{(\theta)}_{X\cup I}]$ is not equivalent to minimizing the $\mathcal{KL}$-divergence on the original index set, in the sense that:*

(i) The two do not have the same numerical value. This is because

$$\mathcal{KL}[Q^{(\theta)}_{X\cup I}||\hat{P}^{(\theta)}_{X\cup I}] = \mathbb{E}_{Q^{(\theta)}_X}\left\{\mathcal{KL}[\,Q^{(\theta)}_{I|X}||P^{(\theta)}_{I|X}]\right\} + \mathcal{KL}[Q^{(\theta)}_X||\hat{P}_X]$$

and $\mathbb{E}_{Q^{(\theta)}_X}\left\{\mathcal{KL}[\,Q^{(\theta)}_{I|X}||P^{(\theta)}_{I|X}]\right\}$ *is not necessarily zero.*

(ii) The two do not necessarily have the same optimal parameters, in the sense that

$$\arg\min_\theta \left\{\mathcal{KL}[Q^{(\theta)}_{X\cup I}||\hat{P}^{(\theta)}_{X\cup I}]\right\} \neq \arg\min_\theta \left\{\mathcal{KL}[Q^{(\theta)}_X||\hat{P}_X]\right\} .$$

Proof. To prove part (i) we apply the chain rule for $\mathcal{KL}$-divergences to $\mathcal{KL}[Q^{(\theta)}_{X\cup I}||\hat{P}^{(\theta)}_{X\cup I}]$ and then note that conditional independence implies that $\hat{P}^{(\theta)}_{I|X} = P^{(\theta)}_{I|X}$. To show that the conditional relative entropy term is not necessarily zero, we can take as an example the case where I and X are finite and all relevant densities exist. Applying Bayes' theorem we have

$$q(f_I|f_X) = \frac{p(f_X|f_I)q(f_I)}{p(f_X)} .$$

Therefore it is clear by inspection that

$$q(f_I|f_X) \neq p(f_I|f_X)$$

in general. Thus the $\mathcal{KL}$-divergence inside the expectation is not necessarily 0.

Given the truth of part (i) it would seem unlikely that (ii) is false, but this does not constitute a proof. It suffices to show a single example where the minima are different which we do in appendix B. □

Since they are important special cases, we detail the consequences of the theorem for the original Titsias argument as corollaries. They can be immediately recovered from the theorem by taking $X = D$ and $I = Z$.

Corollary 4.1. *Proposition 1 is false. That is to say, in general:*

$$\mathcal{KL}\left[Q^{(\theta)}_D||\hat{P}_D\right] \neq \mathcal{KL}\left[Q^{(\theta)}_{D\cup Z}||\hat{P}^{(\theta)}_{D\cup Z}\right]$$

Corollary 4.2. *Proposition 2 is false. That is to say in general:*

$$\arg\min_\theta\left\{\mathcal{KL}\left[Q^{(\theta)}_D||\hat{P}_D\right]\right\} \neq \arg\min_\theta\left\{\mathcal{KL}\left[Q^{(\theta)}_{D\cup Z}||\hat{P}^{(\theta)}_{D\cup Z}\right]\right\}$$

Given the issues with the original motivation in Titsias' paper, where does this leave the method? The answer is that it is still valuable. Although neither proposition 1 or proposition 2 can be used to motivate the use of the $\mathcal{KL}$-divergence (3.1), as already remarked theorem 2.ii does. Thus everything that follows from the use of the $\mathcal{KL}$-divergence (3.1), still applies. This includes Titsias' (2009a) elegant solution for the optimal inducing output solution in the Gaussian likelihood case. Another advantage of using the full $\mathcal{KL}$-divergence $\mathcal{KL}[Q_X||\hat{P}_X]$ as a starting point is that it includes the posterior over new functions values that we may wish to predict as part of the optimization objective.

There is a generalization of the sparse variational framework which allows a more expressive variational distribution, whilst maintaining $\mathcal{KL}[Q_X||\hat{P}_X]$ as its optimization objective. Although it can be understood as a special case of augmentation where the model augmentation is conditionally deterministic, it is clearer to introduce it as a standalone method, which we will do in the next section.

3.5 Variational inducing random variables

In this section, we consider a generalization of the variational inducing point framework derived in section 3.3.3. We call this generalization the *variational inducing random variable framework*. Currently, the main application of this framework is to the rigorous treatment of the variational interdomain inducing framework given in section 3.6.1.

Our starting point is the definition of the variational approximation given in equation 3.14. We replace the projection $\pi_Z : \mathbb{R}^X \mapsto \mathbb{R}^M$ with a general measurable *transformation function* $h_\theta : \mathbb{R}^X \mapsto \mathbb{R}^M$, with parameters θ. The definition of the variational approximation therefore becomes

$$\frac{dQ}{dP}(f) = \gamma(h_\theta(f)) \,. \tag{3.35}$$

We define the inducing random variables f_R by the equation

$$f_R = h_\theta(f) \,. \tag{3.36}$$

Although the inducing random variables depend on the parameters θ we will not show this explicitly in order to keep the notation concise. For any measure μ on the function space of the index set X the measure μ_R will denote the distribution of the inducing random variables for some Borel set A in the standard way

$$\mu_R(A) = \mu_X(h_\theta^{-1}(A)) \,. \tag{3.37}$$

By a similar reasoning to that followed in section 3.3.3, γ may be replaced with the Radon-Nikodym derivative $\frac{dQ_R}{dP_R}$ to give

$$\frac{dQ}{dP}(f) = \frac{dQ_R}{dP_R}(h_\theta(f)) \,. \tag{3.38}$$

The variational approximations of section 3.3.3 are a special case of this approximation with $h_\theta = \pi_Z$. We now give a theorem on the $\mathcal{KL}$-divergence between this approximating process and the posterior process $\hat{P}$.

Theorem 5. *Let the approximating measure be defined by* (3.38) *and the posterior measure be given by* (3.13)*. Then the $\mathcal{KL}$-divergence $\mathcal{KL}[Q||\hat{P}]$ between these processes obeys the relation*

$$\mathcal{KL}[Q||\hat{P}] = \mathcal{KL}[Q_R||P_R] - \mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right] + \log p(Y) \,.$$

Proof. The proof of this result follows a similar route to that of theorem 2.i , replacing f_Z with f_R. □

The main caveat on the application of this theorem is the need to show the measurability of the transformation h_θ. By contrast to the special case $h_\theta = \pi_Z$ which was discussed in section 3.3.3, showing measurability in the general case can be somewhat involved.

In the language of section 3.4, this approximation may be viewed as *conditionally deterministic augmentation*. That is to say, the inducing random variables framework corresponds to an augmentation framework where the augmentation variables are deterministic conditional on the whole function f_X. This result is proven in Appendix C. By contrast to the augmentation framework without the 'deterministic conditional' caveat, theorem 5 tells us that the inducing random variable approximation has $\mathcal{KL}[Q||\hat{P}]$ as its objective function.

For the inducing random variable approximation to be practical, the resulting approximating distribution Q must be tractable along with $\mathcal{KL}[Q_R||P_R]$ and $\mathbb{E}_{Q_D}\left[\log p(Y|f_D)\right]$. It is easier to ensure this if the transformation function h_θ is a *linear functional*, because then f_R will have a Gaussian joint distribution with f_X. An important example of such a linear transformation is the family of interdomain approximations, where h_θ is defined in terms of an integral. This will be the first example in the next section.

3.6 Examples

3.6.1 Variational interdomain approximations

Here we consider the sparse variational interdomain approximation which was suggested but not realized in Figueiras-Vidal and Lazaro-Gredilla (2009) and appeared under the basis of the marginal consistency argument in Alvarez et al (2010). As already mentioned, interdomain random variables are a special case of the inducing random variables framework of the previous section. In particular the transformation h_θ, which in this case is a linear functional, is defined in terms of a set of integrals over the function f. Let $r \in R$ index the inducing random variables f_R so that each value f_r is a real number. Associate with each $r \in R$ a set of variational parameters θ_r. We may then define f_r as

$$f_r = \int_X g(x, \theta_r) f_x \, d\lambda(x) \tag{3.39}$$

where λ is a measure on X with some appropriate σ-algebra. g as a function of its first argument is a λ-integrable function from X to $\mathbb{R}$ for all values of its second argument θ_r.

As a specific example of an interdomain variable (Figueiras-Vidal and Lázaro-Gredilla, 2009), consider the case where the index set X is the real line and g takes the form of a wavelet

$$g(x, \omega_r, t_r, b_r) = \cos\left(\omega_r(x - t_r)\right) \exp\left\{ -\frac{(x - t_r)^2}{2b_r^2} \right\} \tag{3.40}$$

where in this instance $\theta_r = (\omega_r, t_r, b_r)$ is the set of variational parameters. Since as $\omega_r, b_r \to 0$, the function g becomes a Dirac delta function centred on t_r such a variational approximation generalizes the standard sparse method.

Since the intention here is to put the general variational interdomain framework on a firm logical footing, we should also consider the thorny issue of the measurability of the associated transformation h_θ. The existence of separable, measurable, versions of stochastic processes, including most commonly used Gaussian processes, was settled in the work of Doob (1953). It also discusses the conditions necessary to apply Fubini's theorem to expectations of the random variable defined by equation (3.39). The application of Fubini's theorem is essential to the utility of such methods in practice (Figueiras-Vidal and Lázaro-Gredilla, 2009).

Section 3.5 tells us we may correctly optimize the parameters θ of interdomain inducing points, safe in the knowledge that this decision is variationally protected from overfitting and optimizes a well defined $\mathcal{KL}$-divergence objective. The potential for a wide variety of improved sparse approximations in this direction is thus, in our opinion, significant.

3.6.2 Approximations to Cox process posteriors

In this section, we relax the assumption that the data set D is finite, which is necessary to consider Gaussian process based Cox processes. Sparse variational inference for one specific case of this model is considered by Lloyd et al (2015). A Gaussian process based Cox process has the following generative scheme:

$$\begin{aligned} f &\sim \mathcal{GP}(m, K) \\ h &= \rho(f) \\ Y|h &\sim \mathcal{PP}(h)\,. \end{aligned} \tag{3.41}$$

Here $\mathcal{GP}(m, K)$ denotes a Gaussian process with mean m and kernel K. $\rho : \mathbb{R} \mapsto (0, \infty)$ is an inverse link function which is applied pointwise to the function f. $\mathcal{PP}(h)$ is a Poisson process with intensity h and D is a set of points in the original index set X. For example, in a geographical spatial statistics application we might take X to be some bounded subset of $\mathbb{R}^2$. The key issue with the Poisson process likelihood is that it depends not just on those members of X where points where observed but in fact on all points in X. Intuitively the absence of points in an area suggests that the intensity is lower there. Thus $D = X$. The likelihood in question is:

$$\begin{aligned} p(Y|f_D) &= \left(\prod_{y \in Y} h(y) \right) \exp \left\{ - \int_X h(x) dm(x) \right\} \qquad (3.42) \\ &= \left(\prod_{y \in Y} \rho(f(y)) \right) \exp \left\{ - \int_X \rho(f(x)) dm(x) \right\} \end{aligned}$$

where m denotes for instance Lebesgue measure on X. The full X dependence manifests itself through the integral on the right hand side. We will require that the integral exists almost surely. In Lloyd et al (2015) equation (3), the application of Bayes' theorem appears to require a density with respect to infinite dimensional Lebesgue measure, as does the integral in equation (9) of that paper. As pointed out in 3.3.1, such a notion is pathological. This however can be fixed because the more general form of Bayes' theorem in equation (3.12) of this chapter can still be applied

$$\frac{d\hat{P}}{dP}(f) = \frac{\left(\prod_{y \in Y} \rho(f(y))\right) \exp\left\{- \int_X \rho(f(x)) dm(x)\right\}}{\int \left(\prod_{y \in Y} \rho(f(y))\right) \exp\left\{- \int_X \rho(f(x)) dm(x)\right\} dP(f)}\,. \tag{3.43}$$

Note that it is necessary to consider the measurability of the right hand side of this equation, just as it was for the integral transformation (3.39). One route here is to use an almost surely continuous modification of f as discussed by Møller et al (1998). Further, we need to use the $\mathcal{KL}$-divergence (3.10) that is valid in infinite dimensional cases. Defining the approximating distribution Q as in equation (3.15), we can apply the results of section 3.3.3 to obtain:

$$\begin{aligned}\mathcal{KL}[Q||\hat{P}] =& \mathcal{KL}[Q_Z||P_Z] - \sum_{y \in Y} \mathbb{E}_{Q_y}\left[\log h(y)\right] \\ &+ \mathbb{E}_{Q_X}\left[\int_X h(x) dm(x)\right] + \log p(Y)\,. \end{aligned} \tag{3.44}$$

As in section 3.6.1 we will need to check that the conditions for Fubini's theorem apply (Doob, 1953) which gives:

$$\begin{aligned}\mathcal{KL}[Q||\hat{P}] =& \mathcal{KL}[Q_Z||P_Z] - \sum_{y \in Y} \mathbb{E}_{Q_y}\left[\log h(y)\right] \\ &+ \int_X \mathbb{E}_{Q_x}\left[h(x)\right] dm(x) + \log p(Y)\,. \end{aligned} \tag{3.45}$$

For the specific case of ρ used in Lloyd et al (2015) the working then continues as from the result of equation (9) in that paper and the elegant results that follow all still apply. Note that one could combine these Cox process approximations with the interdomain framework and this could be a fruitful direction for further work.

3.7 Conclusion

In this chapter we have elucidated the connection between the variational inducing point framework (Titsias, 2009a) and a rigorously defined $\mathcal{KL}$-divergence between stochastic processes. Early use of the rigorous formulation of $\mathcal{KL}$-divergence to the Gaussian processes for machine learning literature was made by Seeger (2003a; 2003b). Here we have increased the domain of applicability of those proofs by allowing for inducing points that are not data points, and removing unnecessary dependence on Lebesgue measure. We would argue that our proof clarifies the central and elegant role played by Radon-Nikodym derivatives. We apply these theorems to categorize a broad set of cases in the literature where increasing the number of inducing points is guaranteed to improve the approximation quality in terms of $\mathcal{KL}$-divergence. We then consider the case where additional variables are added solely for the purpose of variational inference. We show that marginal consistency is not enough to guarantee that variational inference in an augmented space is equivalent to variational inference in the unaugmented space. This has negative implications for Titsias' original augmentation argument. Instead, we advocate the variational inducing random variable framework, which generalizes sparse approximations whilst maintaining the posterior process as its optimization target. We then show how the extended theory allows us to correctly handle principled interdomain sparse approximations and that we can cope correctly with the important case of Cox processes where the likelihood depends on an infinite set of function points.

It seems reasonable to hope that elucidating the measure theoretic roots of the formulation will help the community to generalise the framework and lead to even better practical results. In particular, it seems that since interdomain inducing points are linear functionals, the theory of Hilbert spaces might profitably be applied here. It also seems reasonable to think given the generality of sections 3.3.3 and 3.5 that other Bayesian and Bayesian nonparametric models might be amenable to such a treatment.

Chapter 4

Scalable Variational Gaussian Process Classification

4.1 Introduction

The Gaussian process framework (Rasmussen and Williams, 2006) has much to recommend it. As Bayesian nonparametric models (Orbanz and Teh, 2010) they simultaneously provide an extremely flexible model whilst allowing principled Bayesian regularization. In the case of regression with the conjugate Gaussian likelihood many of the quantities of interest have analytic expressions in terms of basic linear algebra. Prior knowledge can be incorporated via the kernel. Unknown hyperparameters can be chosen using type-II likelihood maximization or a fully Bayesian treatment.

4.1.1 Non-conjugate likelihoods[1]

A challenge is encountered when we try to go beyond the Gaussian likelihood to a non-conjugate likelihood. To be concrete, consider the class of models

$$f \sim \mathcal{GP}(0, K(\theta)) \tag{4.1}$$

$$y \,|\, d, f \sim h(f(d)), \qquad y, d \in Y, D \text{ i.i.d}\,. \tag{4.2}$$

Here $f : X \mapsto \mathbb{R}$ is a function from the index set X to the set of real numbers $\mathbb{R}$. K is the covariance function $K : X \times X \mapsto \mathbb{R}$, which for marginal consistency must be a positive

[1] Note that some authors reserve the use of the term conjugate for priors only. We prefer to think of conjugacy as a relation that applies (or does not apply) to a likelihood/prior pair.

semi-definite Mercer kernel. We follow the common convention of assuming a zero prior mean function for ease of exposition but everything that follows can be straightforwardly generalized to this case. θ denotes the covariance hyperparameters like signal variance and characteristic length scale. In this chapter we will take point estimates of the hyperparameters, chosen using approximate type-II maximum likelihood. The extension to a Bayesian treatment of these parameters will be discussed in chapter 5. The data comes in N pairs $y, d \in Y, D$. h is some distribution over the possible outputs which depends on the value of the function at the input point. It is easy to include likelihood parameters but these are omitted for brevity.

In this non-conjugate case almost all required quantities require approximate integration and there has been considerable literature devoted to this question. Despite these challenges, the elegant mathematical properties of the underlying prior on functions mean that the Gaussian process framework is still a compelling one in this domain.

4.1.2 Scalability challenges

In the case of regression for a generic covariance function and input data set, the linear algebra necessary for posterior inference requires $\mathcal{O}(N^3)$ computation. Up to about 2000 data points this is not prohibitive with a modern desktop CPU and there are plenty of interesting datasets that fall into this category. However, in modern machine learning, datasets are frequently much larger than this and then the adverse computational scaling becomes prohibitive. Many of the approximation methods that are used for non-conjugate likelihoods inherit this adverse scaling since the underlying prior on functions remains the same.

4.1.3 A specific realization: classification

Many of the insights offered in this chapter apply to a broad range on non-conjugate likelihoods however it will be helpful to work with a specific example, namely Gaussian process classification (GPC). Classification, particularly at scale, is very heavily studied in Statistics and Machine Learning because the basic setup is so common in applications. For binary classification the canonical way to use a Gaussian process prior is first to map the function through an inverse link function g:

$$g : \mathbb{R} \mapsto [0, 1] \tag{4.3}$$

This has the effect of mapping the function from the real line to the unit interval so that it can represent a probability. The next stage is to sample a Bernoulli random variable corresponding to the class conditioned on the value of this contracted function:

$$y \,|\, d, f \sim \mathcal{B}(g(f(d))) \tag{4.4}$$

The importance of classification as an application has led to a significant body of research on $\mathcal{O}(N^3)$ approximate inference for Gaussian processes. See (Nickisch and Rasmussen, 2008) for an excellent review. The wealth of available comparison points makes it a natural starting point for any systematic study of scalable non-conjugate Gaussian process inference. The extension of our work to multiclass classification, which is important in applications, is discussed in section 4.4.

4.1.4 A summary of the challenge

This chapter addresses two challenges in GP inference:

1. Approximate integration in non-conjugate Gaussian process modelling using classification as our main focus.

2. The adverse $\mathcal{O}(N^3)$ scaling with the number of input data points.

4.1.5 An outline for the rest of the chapter

This chapter will next situate our work by reviewing some background literature on addressing the two challenges highlighted (section 4.2). This review will be structured around the key features that we argue a successful method needs. We will then describe our proposed solution to this problem (section 4.3), which has the desired characteristics. This method follows naturally from the considerations of chapter 3, but we have written this chapter to require less technical mathematics. As we will discuss in section 4.4, the method has a natural extension to multiclass classification. Next (section 4.5), we will thoroughly evaluate our proposed method in a series of empirical experiments. The main comparison point is the popular Generalized FITC algorithm of Naish-Guzman and Holden (2008). In these comparison experiments, we find that our method has competitive performance. We then demonstrate that using stochastic variational inference enables us to obtain good results on the Airline delay dataset (which has roughly 5.8 million datapoints) and the MNIST digit classification dataset. Finally, we conclude (section 4.6).

4.2 Background

Here we describe the key components of our proposed solution to the above two challenges, giving a critical review of the history of each aspect.

4.2.1 Non-conjugate approximations for GPC

There is a large literature on approximating the non-Gaussian posteriors associated with Gaussian process classification. See (Nickisch and Rasmussen, 2008) for an excellent review. That paper finds that of the broad range of methods reviewed, Expectation Propagation (EP) (Minka, 2001), when applied to Gaussian process classification (Csató and Opper, 2002; Kuss and Rasmussen, 2005) gives the most accurate posterior mean and covariance predictions as well as the best proxy to the marginal likelihood for hyperparameter optimization. In their review, Nickisch and Rasmussen (2008) also reported that the variational method of Opper and Archambeau (2009) was similar but gave slightly less accurate estimates for both the posterior moments and marginal likelihood. In the implementation used in their comparison paper, the authors also found the variational approximation algorithm required significantly longer to terminate. There is no suggestion in the paper that this optimization issue is insurmountable. From a theoretical standpoint proving that the variational method will always converge is straightforward since it is constructed using a bounded objective function which is then optimized. By contrast whilst EP for GPC is usually *observed* to converge, a proof that this will always occur has so far eluded researchers.

To summarize, Expectation Propagation for this problem can exhibit good empirical performance but some of the desirable theoretical foundation is currently lacking. A lack of theory can obviously make principled generalizations more difficult.

4.2.2 Sparse Gaussian processes

The Sparse Gaussian process framework is a family of methods for approximating computationally expensive full GP inference. One informative way of understanding many of these methods is via the "Unifying View" advocated by Quinoñero-Candela and Rasmussen (2005). Their starting point was the following exact relation for the GP prior:

$$p(f_*, f_D) = \int p(f_*, f_D | f_Z) p(f_Z) df_Z \tag{4.5}$$

where f_* is a finite set of test points and f_D corresponds to the function values at the data points. This is approximated by the density $r(f_*, f_D)$ which is defined in terms of some inducing outputs f_Z

$$p(f_*, f_D) \approx r(f_*, f_D) = \int r(f_*|f_Z) r(f_D|f_Z) p(f_Z) df_Z \,. \tag{4.6}$$

Many sparse Gaussian process approximations correspond to using $r(f_*, f_D, f_Z)$ as an approximation to the prior and then proceeding as if this was the actual prior. These approximations differ in their choice of $r(f_*|f_Z)$ and $r(f_D|f_Z)$. They are all similar in that the additional assumed factorization corresponds to a conditional independence assumption. Under the approximation, the function values at the data points f_D and the function values at the test points f_* are independent conditioned on f_Z. It is this assumption that gives a reduction in the complexity of inference from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$. Two special cases will be particularly relevant here.

Using the nomenclature and insight of Quinoñero-Candela and Rasmussen (Quiñonero-Candela and Rasmussen, 2005) the Deterministic Training Conditional (DTC) approximation (Seeger, 2003a) corresponds to taking

$$r_{\text{DTC}}(f_D|f_Z) = \mathcal{N}(f_D|K_{D,Z}K_{Z,Z}^{-1}f_Z, 0) \tag{4.7}$$

$$r_{\text{DTC}}(f_*|f_Z) = p(f_*|f_Z) \tag{4.8}$$

and the Fully Independent Training Conditional (FITC) approximation of Snelson and Ghahramani (2005) corresponds to taking

$$r_{\text{FITC}}(f_D|f_Z) = \mathcal{N}(f_D|K_{D,Z}K_{Z,Z}^{-1}f_Z, \text{Diag}[K_{D,D} - K_{D,Z}K_{Z,Z}^{-1}K_{Z,D}]) \tag{4.9}$$

$$r_{\text{FITC}}(f_*|f_Z) = p(f_*|f_Z) \,. \tag{4.10}$$

Here Diag is an operator that takes a square matrix and returns a square matrix with only the diagonal elements of the input. We see that DTC and FITC differ in their training conditional variance.

4.2.3 Subset of data methods and subset of data inducing methods

A subset of data method chooses some subset M of the data points and uses this cheaper posterior as a proxy for the real posterior. The simplest method is to choose the points at random. There are more principled methods for choosing data points for instance in the case of the Informative Vector Machine (Lawrence et al., 2003), the points are chosen using an information theoretic criterion. Clearly the point selection criterion itself needs to be tractable. However sophisticated a selection method is employed, Quinoñero-Candela and Rasmussen (2005), section 3, points out that such methods will tend to overestimate uncertainty relative to the exact posterior. This is because with fewer data points we will tend to be more uncertain about the underlying function. We would add that in a similar way they will tend to give marginal likelihood estimates that are less peaked around the maximum likelihood hyperparameter values than they should be.

A subset of data inducing method selects points from the input data set and treats them as *inducing points* rather than *data points*. Deferring discussion of the exact selection criterion to the next section, all such methods have in common that they require the solution or approximate solution of a difficult combinatoric optimization problem. As described by Snelson and Ghahramani (2005) section 1, this leads to difficulty in simultaneous optimization of continuous parameters such as kernel hyperparameters. The solution proposed by these authors is that the inducing points should be chosen from the continuous underlying index set X. This raises the question as to what happens if the underlying index set is in fact discrete (e.g graphs). This issue is still a challenging one. For the most recent progress on this problem see (Cao et al., 2013). In the majority of applications, however, the covariates are continuous or have a reasonable continuous relaxation. In such applications continuous inducing point optimization has proved essential to good results.

Notable for our discussion is the work of Seeger (2003b), Titsias (2009a) section 3.1 and Chai (2012) who employ the variational criterion for discrete inducing point selection. In the last of these papers it is not clear why the authors did not use continuous optimization. We speculate this may be a symptom of the residual confusion around what the optimization objective actually was and which inducing points were admissible that we resolved in chapter 3.

4.2.4 A coherent variational framework

In 2009 an alternative view emerged that did not fit into the prior approximation framework proposed by Quinoñero-Candela and Rasmussen (2005). Titsias (2009a) proposed fitting the posterior approximation using a $\mathcal{KL}$-divergence criterion. In chapter 3 we showed that this criterion can be interpreted as the $\mathcal{KL}$-divergence between processes and argued that this has some benefits over Titsias' original 'augmentation' argument. The approach may therefore be interpreted as defining a variational family $\mathcal{Q}$ of approximating processes Q and choosing one according to $\mathcal{KL}$-divergence minimization. For fixed hyperparameters and inducing points, regression with a Gaussian likelihood in this framework has the same predictive mean and variance as the DTC approximation (Titsias, 2009a).We postpone a more detailed discussion of this to section 5.2 on 'variational potentials'.

Those who adopt the prior approximation view typically treat the inducing point positions as covariance hyperparameters and optimize them using type-II maximum likelihood. The DTC prior approximation has the following marginal likelihood

$$\log r_{DTC}(Y|\theta) = \log \mathcal{N}(Y|\sigma^2 I + K_{D,Z}K_{Z,Z}^{-1}K_{Z,D}) \tag{4.11}$$

where σ^2 is the noise variance associated with the Gaussian likelihood and the notation convention for kernel matrices is the same as the one introduced in section 1.2.3. For regression Titsias showed that the variational lower bound on the exact marginal likelihood takes the form

$$\mathcal{L} = \log \mathcal{N}(Y|\sigma^2 I + K_{D,Z}K_{Z,Z}^{-1}K_{Z,D}) - \frac{1}{2\sigma^2}\operatorname{Tr}\left(K_{Z,Z} - K_{D,Z}K_{Z,Z}^{-1}K_{Z,D}\right) \tag{4.12}$$

which differs from the DTC approximation to the marginal likelihood in that it contains a second 'trace' term. It is precisely this additional term that changes the inducing point positions to be *variational parameters* instead of *hyperparameters*. Put another way, in the variational framework the inducing point positions will be chosen to move the approximation closer to the posterior. In the prior approximation view the inducing point positions are chosen to maximize the likelihood of the data under the approximate model. In the latter case there is no guarantee that increasing the number of inducing points will bring the approximation closer to the posterior and one might expect problems as the number of inducing points increases. In fact Snelson and Ghahramani (2005) and later Naish-Guzman and Holden (2008) observe that FITC has heteroscedastic behaviour which is not present in the original model. This has sometimes even been argued to be a benefit of such an approach citing better test likelihoods on certain datasets. This seems, in our opinion, to have been making a virtue of a necessity. Of course if this was indeed still a necessity then making a virtue of it would be a sensible and pragmatic thing to do! However now that methods exist which do ensure that extra computation will bring the approximation closer to the posterior we would argue that the right way to introduce extra modelling assumptions, if they are genuinely wanted, would be to put them into the prior and likelihood in the standard Bayesian way.

In this thesis we have generally avoided the 'lower bound' style derivations that are commonly encountered in other works. This is deliberate because we feel that such arguments can obscure the crucial question of the form that the slack in the bound takes. For all the applications we consider we want a method that not only gives a tight bound $\mathcal{L}$ on the marginal likelihood but also for fixed hyperparameters gives a good approximation to the posterior. This second criterion is difficult to guarantee unless the slack in the bound takes the form of a $\mathcal{KL}$-divergence from an approximating distribution to the posterior

$$\mathcal{L} + \mathcal{KL}[Q||\hat{P}] = \log p(Y)\,. \tag{4.13}$$

Some of the bounds used in the literature are known to have this property and some of them are not. As we have discussed in some detail the bound (4.3) does have this property. Although it would be interesting to consider necessary theoretical conditions for such a relation existing, we have not yet pursued this avenue. In the current context, two important examples of methods that are not known to obey the relation in equation (4.13) and that I conjecture do not, are the non-Gaussian likelihood approximation schemes of Hensman et al (2013), section 3.4 and the 'two stage' mean field approach of Hensman et al (2015b), section 3. The approximation of Hensman et al (2013), section 3.4 was suggested as a generalization of the main regression method of the paper to non-Gaussian likelihoods, but not implemented. It has since been found to give poor empirical performance (Owen Thomas, personal communication). The 'two stage' mean field approach of Hensman et al (2015b), section 3, was found in that paper to be poorly calibrated relative to other relevant methods. Of course, an easy way to guarantee that a relation like (4.13) is obeyed is to use $\mathcal{KL}[Q||\hat{P}]$ as the starting point. It is this approach that we investigate in section 4.3, where the method has a relatively clear derivation and good empirical performance.

4.2.5 Stochastic variational inference

For the largest datasets, even reducing the computational intensity to $\mathcal{O}(N)$ per iteration is still prohibitively slow. Stochastic variational inference (Hoffman et al., 2013) (SVI) has proved to be a particularly impactful and successful approach to scalable Bayesian inference. The essential idea behind SVI is that in following the gradient of the variational ELBO a noisy but unbiased estimate obtained from minibatches is often sufficient to give a good update of the variational parameters. Robbins and Monro (1951) characterized conditions on the stepsize under which a first order stochastic gradient method would converge and there has been a significant literature on this topic since. These elegant convergence properties are essential to the success of the algorithm. For machine learning algorithms in general one could envisage many ways to use a minibatch as an approximation for a full pass over the data but relatively few of them would have this desired convergence guarantee.

The coherent variational inference framework provided by Titsias (2009a) raised the possibility that the all free parameters could be updated in such a way and maintain convergence. An obstacle existed however in this sense that Titsias' variational approximation could not

Table 4.1 A summary of the most relevant existing approaches. The presence of the question mark for (Hensman et al., 2013) is explained in section 4.2.4. SOD stands for 'subset of data'.

Reference	**Non-conjugate**	**Sparse**	**Variational**	**Beyond SOD**	**SVI**
Snelson and Ghahramani (2005)	✗	✓	✗	✓	✗
Titsias (2009a)	✗	✓	✓	✓	✗
Hensman et al. (2013)	✗	✓	✓	✓	✓
Hensman et al. (2013) (suggested)	✓	✓	?	✓	✓
Lawrence et al. (2003)	✓	✓	✗	✓	✗
Chai (2012)	✓	✓	✓	✗	✗
Naish-Guzman and Holden (2008)	✓	✓	✗	✓	✗
Seeger (2003a)	✓	✓	✓	✗	✗
This Work	✓	✓	✓	✓	✓

be written as a sum over data terms - a requirement necessary to apply minibatches. This challenge was solved by Hensman et al (2013) who recast the problem in a way amenable to stochastic inference.

4.2.6 Bringing the components together

In order to help the reader keep track of the relevant existing work we have summarized which papers possessed which of the features in table 4.1. Whilst we think that in light of the complexity of the existing literature such a summary is helpful, we would caution the reader against the conclusion that achieving such an ideal combination of features is a matter of easily patching together existing approaches.

Prior to this work the most commonly adopted method for approximating Gaussian process classification at scale would be to take the approximate prior of FITC and then perform EP to deal with the non-conjugate observations. Careful book keeping keeps the computational scaling to $\mathcal{O}(NM^2)$. This is essentially the approach taken by Naish-Guzman and Holden (2008) with their Generalized FITC approximation (GFITC). Whilst one would hope to inherit the good classification performance of EP using this method one will also inherit the less principled FITC approach to sparsity. By contrast, as we shall see, the approach we adopt solves the inducing point selection problem, the non-conjugacy and the mini-batch requirement all using the same unifying variational principle. Taking a step back from all this technical detail our approach has enabled us to give good results for approximate GPC on much larger and more challenging classification datasets then all previous published work on the topic.

4.3 Sparse variational Gaussian process classification

In this section we describe our approach to the sparse GPC problem. The derivation we give here is similar to that given in chapter 3. It is sufficiently important for understanding our approach in this chapter that we risk some repetition. The exposition here is less formal and general but is hopefully more intuitive. We assume that the data points D, the inducing points Z and test set $*$ are disjoint for ease of exposition but this can be relaxed as in chapter 3.

The Kolmogorov extension theorem, introduced in section 1.1.1, tells us that specifying the finite dimensional marginal distributions of a process is sufficient to specify most relevant questions about the whole process. Here we work in terms of the densities of these finite marginal distributions.

The prior process can be decomposed in several ways using the chain rule:

$$
\begin{aligned}
p(f_*, f_D, f_Z) &= p(f_*|f_D, f_Z)p(f_D|f_Z)p(f_Z) &\quad (4.14)\\
&= p(f_*|f_D, f_Z)p(f_Z|f_D)p(f_D) &\quad (4.15)
\end{aligned}
$$

The posterior process has similar decompositions:

$$
\begin{aligned}
p(f_*, f_D, f_Z|Y) &= \frac{p(Y|f_D)p(f_*|f_D, f_Z)p(f_D|f_Z)p(f_Z)}{p(Y)} &\quad (4.16)\\
&= \frac{p(Y|f_D)p(f_*|f_D, f_Z)p(f_Z|f_D)p(f_D)}{p(Y)} &\quad (4.17)
\end{aligned}
$$

Any function values for inputs that are not data inputs are conditionally independent of the output data Y given the data function values f_D. This means Bayes' theorem applies 'locally' to the data function values f_D and that the distribution for the rest of the function values are then obtained by multiplying by the prior conditional distributions

$$
p(f_*, f_D, f_Z|Y) = p(f_*|f_D, f_Z)p(f_Z|f_D)p(f_D|Y)\,. \quad (4.18)
$$

Now we define our approximating family $\mathcal{Q}$. A member of the family is fully specified by the choice of inducing points Z and a distribution over the inducing point function values $q(f_Z)$ which is taken from some family of distributions $\mathcal{Q}_Z$. The distribution is given by:

$$
q(f_*, f_D, f_Z) = p(f_*|f_D, f_Z)p(f_D|f_Z)q(f_Z) \quad (4.19)
$$

Note that we multiply by the prior conditionals to obtain the full joint distribution from $q(f_Z)$. This is why we use the term 'prior conditional matching' to describe this type of variational inference, as in section 2.2.4. The relationship between the variational approximating distribution and the prior distribution can best be seen by comparing equations (4.14) and (4.19): the prior density $p(f_Z)$ has been replaced by $q(f_Z)$. The relationship between the variational approximating distribution and the posterior distribution is made clear by comparing equations (4.18) and (4.19). We see the similarity if we both interchange the roles of f_Z and f_D and substitute $q(f_Z)$ for $p(f_D|Y)$. In this sense $q(f_Z)$ may be thought of as a local 'pseudo posterior' where both the 'pseudo inputs' and the pseudo posterior itself will be tuned variationally to match the exact posterior as closely as possible.

In chapter 3, theorem 2.i , we showed that the $\mathcal{KL}$-divergence $\mathcal{K}$ from the approximating process to the posterior process is given by

$$\mathcal{K} = \mathcal{KL}[q(f_Z)||p(f_Z)] - \mathbb{E}_{q(f_D)}\left[\log p(Y|f_D)\right] + \log p(Y)\,. \tag{4.20}$$

Since for classification the likelihood factorizes, the expression simplifies

$$\mathcal{K} = \mathcal{KL}[q(f_Z)||p(f_Z)] - \sum_{y,d \in Y,D} \mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] + \log p(Y)\,. \tag{4.21}$$

In passing, we note for those readers who find it more intuitive, that this equation can also be arranged to give an evidence lower bound (ELBO) :

$$\begin{aligned}\log p(Y) &= \mathcal{K} + \sum_{y,d \in Y,D} \mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] - \mathcal{KL}[q(f_Z)||p(f_Z)] \\ &\geq \sum_{y,d \in Y,D} \mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] - \mathcal{KL}[q(f_Z)||p(f_Z)] \end{aligned} \tag{4.22}$$

where we have applied the non-negativity of $\mathcal{KL}$-divergences. We also note that as variational methods the proposed training objectives of Seeger (2003b) and Chai (2012) are similar to the bound (4.22). We refer the reader to section 4.2 and table 4.1 for a comparison with our own method.

In this chapter, we will take the variational family of inducing point distributions $\mathcal{Q}_Z$ to be the set of all non-degenerate multivariate normal distributions on $\mathbb{R}^M$

$$q(f_Z) = \mathcal{N}(f_Z|m,\Sigma)\,. \tag{4.23}$$

Target	$q(f_T)$	$p(f_T)$	$p(f_T\|Y)$
Mean	$\mathcal{O}(\|T\|M + M^3)$	0	$\mathcal{O}(\|T\|N + N^3)$
Covariance	$\mathcal{O}(\|T\|^2 M + \|T\|M^2 + M^3)$	$\mathcal{O}(\|T\|^2)$	$\mathcal{O}(\|T\|^2 N + \|T\|N^2 + N^3)$
Sampling	$\mathcal{O}(\|T\|^3 + \|T\|^2 M + \|T\|M^2 + M^3)$	$\mathcal{O}(\|T\|^3)$	$\mathcal{O}(\|T\|^3 + \|T\|^2 N + \|T\|N^2 + N^3)$

Table 4.2 A comparison of the time complexity of various computational operations on the density of a generic function f_T, where T are not inducing points, for the variational approximation, the prior and a Gaussian posterior. $|T|$ is the number of points in T, N is the number of data points and M is the number of inducing points. The prior mean function is assumed to be zero.

This is a subset of all possible distributions on $\mathbb{R}^M$ and is not in general optimal in the sense of minimizing the $\mathcal{KL}$-divergence. The efficacy of this Gaussian assumption will be probed in chapter 5 where we investigate a method that relaxes this assumption.

It will be constructive to consider the approximating density on a general set T of size $|T|$, with the only assumption being that T is finite and disjoint from the inducing input set. We can use the general insights gained to consider the case where T is the data set D or a test set $*$, etc. Using the expression $q(f_T) = \int_{\mathbb{R}^M} p(f_T|f_Z)q(f_Z)df_Z$ and a common Gaussian identity (Appendix D.1) we can marginalize to obtain the implied joint approximating density for f_T

$$q(f_T) = \mathcal{N}(f_T|K_{T,Z}K_{Z,Z}^{-1}m, K_{T,T} - K_{T,Z}K_{Z,Z}^{-1}K_{Z,T} + K_{T,Z}K_{Z,Z}^{-1}\Sigma K_{Z,Z}^{-1}K_{Z,T})\,. \quad (4.24)$$

From this density we can inspect the linear algebra to ascertain the time complexity. The results are summarized in table 4.2 along with the corresponding requirements for the prior and a Gaussian posterior.

We need to parameterize the normal variational distribution $q(f_Z)$. We parameterize the variational mean m directly- the challenge arises in parameterizing Σ. Ideally we would like a parameterization that is unconstrained and that guarantees Σ will be positive definite. Both criteria are fulfilled by taking the variational parameters to be a lower triangular matrix L where

$$LL^T = \Sigma\,. \quad (4.25)$$

Σ is then positive semi-definite by construction. The degenerate cases with zero eigenvalues will be avoided in practice because they have an infinite $\mathcal{KL}$-divergence. Thus we can assume that our parameterization is effectively positive definite. This is of course closely related to the Cholesky decomposition of the positive definite matrix Σ. Generally, however the Cholesky decomposition is assumed to have strictly positive diagonal entries which makes it unique. Here we will relax this assumption, gaining an unconstrained optimization at the expense of uniqueness. This method is found to work well in practice.

We will now discuss in more detail each term on the right hand side of (4.21). The log marginal likelihood $\log p(Y)$ is considered intractable. In the non-conjugate case this would be because it is a difficult, non-analytic, high dimensional integral. In the Gaussian regression case the integral would be analytically tractable in terms of common linear algebra quantities but would require $\mathcal{O}(N^3)$ computation. Either way, as is common in variational inference, we exploit the fact that this term is constant with respect to the variational parameters and thus can simply ignore it for the purpose of choosing the variational approximation.

Next, we consider the prior $\mathcal{KL}$-divergence term. Since the prior and approximating distributions over f_Z are both multivariate normal the $\mathcal{KL}$-divergence between them is analytically tractable

$$\mathcal{KL}[q(f_Z)||p(f_Z)] = \frac{1}{2}\left\{\mathrm{Tr}(K_{Z,Z}^{-1}\Sigma) + m^T K_{Z,Z}^{-1} m - M + \log\left(\frac{\det K_{Z,Z}}{\det \Sigma}\right)\right\} \tag{4.26}$$

This term and its relevant gradients can be evaluated in $\mathcal{O}(M^3)$.

Finally we deal with the sum over likelihood terms. First we need the one dimensional marginal distribution $q(f_d)$ for each $d \in D$. This is a special case of equation (4.24) and computing the mean and variance would thus individually require $\mathcal{O}(M^3)$ computation. However, we can share the $\mathcal{O}(M^3)$ between data points, so computing all N means and variances actually requires $\mathcal{O}(M^3 + NM^2)$. Once we have computed the moments of the marginal distributions we need to evaluate the expectation

$$\int_{\mathbb{R}} q(f_d) \log p(y|f_d) df_d \tag{4.27}$$

for each data point. As a one dimensional quadrature, this is much easier than the high dimensional integrals encountered in non-analytic marginal likelihoods. There has been a recent fashion for approximating such integrals and their gradients with unbiased stochastic estimates. The simplest such method is a näive Monte Carlo estimate that samples from $q(f_d)$. A good reference among many is that of Titsias and Lázaro-Gredilla (2014). The Robbins and Monro (1951) convergence theorem will tolerate variation in the sampled gradients, as long as they are unbiased. However in our experiments we found that Gauss-Hermite quadrature performed better for the same number of function evaluations. Technically, since our functions do not necessarily fulfil the requirements necessary for zero quadrature error, this may introduce bias into the optimization. In reality, though, the bias is negligible in that orders of magnitude more Monte Carlo function evaluations than Gauss-Hermite evaluations would be required to observe the bias beneath the Monte Carlo noise. This insight was later extended to two dimensions and quantified in the supplementary material of Saul et al (2016). We also will need the gradients of the likelihood quadrature. We find that the most numerically stable method is to exploit the identities (D.6) and (D.7) described in Appendix D.2.

Bringing all the terms in (4.21) together, evaluation of the optimization objective and its gradients requires $\mathcal{O}(M^3 + NM^2)$ per iteration which is less than the $\mathcal{O}(N^3)$ of non-sparse approximations when $M \ll N$. However if N is sufficiently large, then even this reduction may not be sufficient since each optimization step requires evaluating the gradient of a likelihood quadrature for each dataset. However, there will often be significant redundancy in computing these gradient terms, and in these circumstances a minibatch sample of the sum will give a representative and unbiased estimate of the gradient. The convergence of simple stochastic gradient descent is then again covered by the theory of Robbins and Monro (1951), although we use the more modern AdaDelta algorithm (Zeiler, 2012) which maintains an online estimate of the second order curvature of the problem. The ease with which it is possible to perform this stochastic variational inference (Hoffman et al., 2013) once the approximation problem has been cast in terms of equation (4.21), is a big advantage of this method relative to other possible approaches to scaling approximate inference in GPs.

Once the training phase is complete we will want to make predictions on hold out data. If we consider a single test input $*$ and its corresponding function value f_* then the necessary predictive density is given by:

$$q(y_*) = \int_{\mathbb{R}} p(y_*|f_*)q(f_*)df_* \,. \tag{4.28}$$

Note that in contrast to the quadrature in equation 4.27 we do not take the log of the likelihood inside the expectation. For the probit classification likelihood this integral is tractable. For other likelihoods we may need to again use Gauss-Hermite quadrature. If we have l different input points in the test set $*$ it can be costly to obtain a correlated density $q(f_*)$ since this is again a special case of equation (4.24), with time complexity described by table 4.2. Consequently, computing a correlated predictive density $q(Y_*)$ also scales badly as l increases. In all experiments for the classification case we therefore simplify predictions by treating them independently. The time complexity then scales as $\mathcal{O}(M^3 + lM^2)$.

4.4 Multiclass classification

We now consider the extension to multiclass classification. Consider the task of classifying J different classes. The model we adopt here is:

$$f^{(j)} \sim \mathcal{GP}(0, K) \qquad j = 1 \ldots, J \;\; \text{i.i.d} \tag{4.29}$$

$$y|d, \mathbf{f}_d \sim \text{Cat}(\mathcal{S}\{\mathbf{f}_d\}) \;\; y, d \in Y, D \tag{4.30}$$

Here Cat is a categorical distribution. $\mathbf{f}_d = (f_d^{(j)})_{j=1}^J$ is the vector of relevant function values. The function $\mathcal{S}$ is a mapping from $\mathbb{R}^J$ to the relevant probability simplex.

We assume that the variational approximation factorizes between classes. The efficacy of this assumption will be discussed in chapter 5 when we exhibit a method that relaxes it. We can follow a derivation similar to the one in section 4.3 to obtain the $\mathcal{KL}$-divergence

$$\mathcal{K} = \sum_{j=1}^{J} \mathcal{KL}[q(f_Z^{(j)})||p(f_Z^{(j)})] - \sum_{y,d \in Y,D} \mathbb{E}_{q(\mathbf{f}_d)} \left[\log p(y|\mathbf{f}_d)\right] + \log p(Y)\,. \tag{4.31}$$

The prior $\mathcal{KL}$-divergence terms can be evaluated much as before. A new challenge is encountered in evaluating the variational expectations of the log likelihood because the quadrature is now multidimensional. Consider for instance choosing the 'softmax' function in the role of the function $\mathcal{S}$

$$\mathcal{S}_{\text{softmax}}(\mathbf{f}_d)_i = \frac{\exp(f^{(i)})}{\sum_{j=1}^{J} \exp(f^{(j)})} \tag{4.32}$$

We could also call this the multiclass generalization of the logit likelihood. Chai (2012) goes in this direction and uses a complex set of local variational bounds to approximate the integral. We found it much simpler algorithmically to take a different approach, using instead the robust max function (Girolami and Rogers, 2006)

$$\mathcal{S}_{\text{robust}}^{(\epsilon)}(\mathbf{f}_d)_i = \begin{cases} 1 - \epsilon, & \text{If } i = \arg\max(\mathbf{f}_d) \\ \frac{\epsilon}{J-1}, & \text{otherwise} \end{cases} \tag{4.33}$$

This likelihood correspond to first taking the class to be the class label of maximal function but then with probability ϵ picking one of the other classes uniformly at random. The parameter ϵ serves two purposes. Firstly if it were zero then the variational expectations would be infinite, involving as they do the logarithm of zero. Secondly, as the name suggests, the parameter gives a degree of robustness to outliers. The addition of white noise to the latent function results in multinomial probit like behaviour (Girolami and Rogers, 2006). A big benefit of choosing the robust max function is that the requisite variational expectation is analytically tractable in terms of the normal cdf and a one dimensional quadrature. We have the following relation

$$\int_{\mathbb{R}^J} q(\mathbf{f}_d) \log(\mathcal{S}^{(\epsilon)}_{\text{robust}}(\mathbf{f}_d)_y) d\mathbf{f}_d = \log(1-\epsilon)S + \log\left(\frac{\epsilon}{J-1}\right)(1-S) \tag{4.34}$$

where S is interpretable as the probability that the function value corresponding to the observed class y is larger than the other function values at that point. Utilizing some basic manipulation and the independence of the variational distribution, the corresponding integral can be simplified

$$\begin{aligned}
S &= \mathbb{E}_{\mathbf{f}_d \sim \mathcal{N}(\mathbf{f}_d|\mu, C)}\left[\mathbb{1}(\mathbf{f}_d^{(y)} > \mathbf{f}_d^{(i)}\ \forall i \neq y)\right] && (4.35)\\
&= \mathbb{E}_{\mathbf{f}_d \sim \mathcal{N}(\mathbf{f}_d|\mu, C)}\left[\prod_{i\neq y} \mathbb{1}(\mathbf{f}_d^{(y)} > \mathbf{f}_d^{(i)})\right] && (4.36)\\
&= \mathbb{E}_{\mathbf{f}_d^{(y)} \sim \mathcal{N}(\mathbf{f}_d^{(y)}|\mu^{(y)}, C^{(y)})}\left\{\prod_{i\neq y} \mathbb{E}_{\mathcal{N}(\mathbf{f}_d^{(i)}|\mu^{(i)}, C^{(i)})}\left[\mathbb{1}(\mathbf{f}_d^{(y)} > \mathbf{f}_d^{(i)})\right]\right\} && (4.37)\\
&= \mathbb{E}_{\mathbf{f}_d^{(y)} \sim \mathcal{N}(\mathbf{f}_d^{(y)}|\mu^{(y)}, C^{(y)})}\left[\prod_{i\neq y} \phi\left(\frac{\mathbf{f}_d^{(y)} - \mu^{(i)}}{\sqrt{C^{(i)}}}\right)\right] && (4.38)
\end{aligned}$$

Here, μ and C correspond to the variational mean and variance of the data point function vector $\mathbf{f}_d$. ϕ corresponds to the standard normal cumulative density function. This one dimensional integral can be evaluated using Gauss-Hermite quadrature.

4.5 Experiments

4.5.1 A critical comparison of the FITC methods and VFE methods for regression data.

In this section we give experimental evidence for the benefits of a coherent variational family which we advocated in section 4.2.4. We do this by making a critical comparison of the Titsias variational free energy (VFE) method (2009a) with the FITC method of Snelson and Ghahramani (2005). Before moving on to non-conjugate approximations, we choose to investigate the case of regression, to remove one layer of complexity from the analysis.

In section 3.3.4 we showed that for a general class of variational methods with fixed kernel hyperparameters that includes the Titsias method for regression, increasing the number of inducing points would not increase the $\mathcal{KL}$-divergence. The Titsias regression method has a zero $\mathcal{KL}$-divergence solution when the number of inducing points equals the number of data points. Further, this optimal solution will give predictions that exactly match the true posterior. The situation is a little more complicated if, as is usually done in practice, the kernel hyperparameters are allowed to vary. The $\mathcal{KL}$-result tells us that the evidence lower bound becomes increasingly tight as we increase the number of inducing points until there is no slack in the bound.

Let us contrast this to the FITC method for regression of Snelson and Ghahramani (2005). As pointed out in the original paper the FITC approximation to the marginal likelihood is correct when the inducing inputs are equal to the true inputs. Further, when the inducing points are in these positions the predictions will be exactly those of the true posterior. We shall refer to this as the 'perfect' solution. But is this solution stable? I.e is it a global or even local minimum of the minimization objective? Here we show empirically that the perfect solution is not a stable stationary point of the FITC optimization objective. That is to say, in practice optimizing the FITC objective function can move us away from the perfect solution even if we initialize there.

To do this we initialize both the VFE and FITC methods at the 'true posterior' solution and then optimize them for a considerable length of time. Initializing at the perfect solution deals with the common critique of such experiments that bad performance is due to a 'poor local optimum' that could allegedly be overcome with better initialization and optimization. Here the poor solutions attained for FITC will have demonstrably lower values of the FITC minimization objective than the perfect solution and the pathology will therefore not be a result of poor initialization.

The experiments in this section where all done using GPFlow (Chapter 6). We use the Snelson dataset (2005) also discussed in section 1.2.3. We take a quarter of the original points as training data. We use an RBF kernel. We use a point estimate of the kernel variance, kernel length scale and noise variance which are chosen by type-II maximum likelihood. The predictions from this exact model are shown in figure 4.1. Figure 4.2 shows the results of the sparse experiments. The Variational Free Energy method stays at the perfect solution as is required by theory. We attribute any small deviation to numerical considerations in the code. The FITC algorithm initially stays near the perfect solution but, proceeding in relatively sudden jumps, it moves away from this solution. The predictive plot shows that

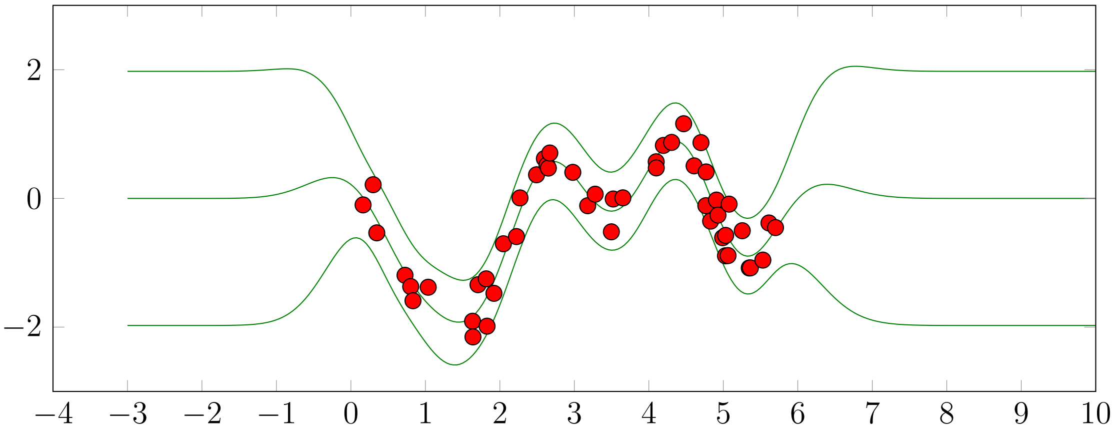

Fig. 4.1 The version of the Snelson dataset used for our experiments. The red points show the training data. The green lines show the posterior mean and two standard deviation credible regions for the observations using a non-sparse model. This exact model, which is described in the text, serves as a reference point for comparison with two sparse approximations.

this solution gives a fairly reasonable value for the predictive mean but a distorted predictive variance. This optimization plot shows that this poor predictive variance has a significant negative effect on the hold out negative log likelihood. Looking at the middle row it appears that the effect of following the FITC objective is to cause the inducing inputs to be placed on top of one another. We speculate that the fact that the initial gradient that carries the FITC approximation away from the perfect solution is very small has meant that this effect not always been fully observed or fully understood in the past.

4.5.2 Demonstrative data and timed comparison

In this experimental section we performed timed binary classification experiments on the Image and Banana datasets (Rätsch et al., 2001), comparing the proposed variational algorithm to the Generalized FITC method with a variety of inducing point numbers.

The Image dataset has 1300 training points, 1000 test points and 18 dimensional covariates. The Banana dataset has 400 training points, 4900 test points and only two dimensional covariates, which makes it easy to visualize the results.

For experiments using the proposed variational method, which we refer to as KLSp, we used code which extended GPy (2012). The inducing inputs were initialized using K-means clustering on the input data. We used the L-BFGS-B optimizer (Zhu et al., 1997). We found that freezing the covariance hyperparameters during an initial period of optimization helped this method find a better optimum.

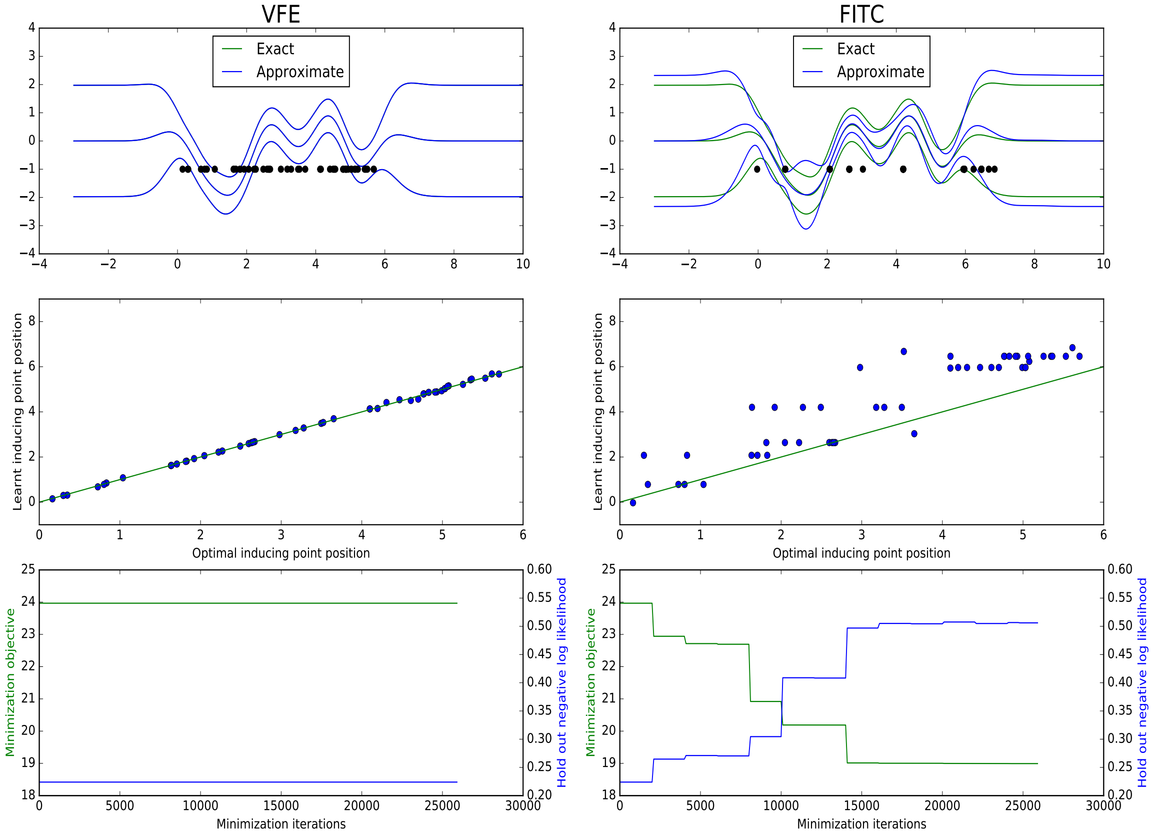

Fig. 4.2 Comparison of FITC and VFE methods to the exact solution on the Snelson dataset in the case $M = N$. The left column corresponds to VFE and the right column corresponds to FITC. Top row: Predictions of the two approximations after minimization of the approximation objective. Middle row: A perfect solution may be attained by keeping the inducing inputs where they were initialized at the data inputs. This row shows the initial inducing point position against the inducing point position after optimization. Bottom row: The approximation minimization objective and the negative hold out log likelihood as optimization proceeds.

The popular Generalized FITC (GFITC) method (Naish-Guzman and Holden, 2008), which we compared to, combines the FITC approximation of the covariance with EP to deal with the non-conjugate likelihood. For the code we used the GPML toolbox (Rasmussen and Nickisch, 2010) which is generally regarded as amongst the best implementations of this algorithm. Again we used K-means clustering to initialize the inducing points and optimized using L-BFGS-B.

For both algorithms on both datasets we recorded the hold out log likelihood and classification accuracy as optimization progressed. We also recorded the corresponding CPU and wall clock times. We report only wall clock time because we found no significant qualitative difference between these two timing metrics. It is difficult to perfectly account for constant factors and levels of software optimization when comparing wall clock time of algorithms, particularly since software packages are being continually improved. The relative times should be viewed as a snapshot but can still be useful. The benefit of recording the whole timing curve rather than recording simply the value of these accuracy metrics and compute time at the end of optimization is that the latter experimental setup can be very sensitive to optimization termination criteria. In many cases performance very close to the totally optimized performance can be achieved in a small fraction of the time. Using the entire time/accuracy curve allow us to characterize the full trade off. In particular we are able to find which algorithms are on the 'efficient frontier' - the curve traced out by the best accuracy achievable for any of the studied algorithms at a given time. For the Banana dataset we ran the optimization for a day.

Since the Banana dataset is two dimensional it was possible to visualize the final classifiers by recording their learnt parameters. As we shall see, there is some evidence that GFITC gets worse for very long runs. Giving the benefit of the doubt to GFITC, we plotted an 'early stopped' classifier state in the visualizations that follow.

The results for the timed experiments are shown in figures 4.3 and 4.4. In both of the timed plots, the effect of the hyperparameter freezing period on KLSp can be seen as an early decrease in the gradient of the hold out metrics, which then suddenly start changing again. In both timed experiments the efficient frontier is entirely made up of the variational KLSp algorithms. For both timed experiments, there is a marked difference in the performance between the two algorithms for small numbers of inducing points ($M = 4$). In the long timed run Banana experiment (figure 4.3), for larger number of inducing points it can be seen that the hold out negative log probability comes back up for GFITC but not KLSp. We suggest that this is explained by similar effects to those observed in section 4.5.1. For the Image dataset (figure 4.4) the large M behaviour is broadly similar for GFITC and KLSp.

Table 4.3 Comparison of the performance of the proposed KLSp method and Generalized FITC on benchmark datasets. In each case we show the median hold out negative log probability and the 2σ confidence intervals.

Dataset	N\P	KL	EP	EPFitc M=8	EPFitc M=30%	KLSp M=8	KLSp M=30%
thyroid	140\5	$.11 \pm .05$	$.11 \pm .05$	$.15 \pm .04$	$.13 \pm .04$	$.13 \pm .06$	$.09 \pm .05$
heart	170\13	$.46 \pm .14$	$.43 \pm .11$	$.42 \pm .08$	$.44 \pm .08$	$.42 \pm .11$	$.47 \pm .18$
twonorm	400\20	$.17 \pm .43$	$.08 \pm .01$	Large	$.08 \pm .01$	$.08 \pm .01$	$.09 \pm .02$
ringnorm	400\20	$.21 \pm .22$	$.18 \pm .02$	$.34 \pm .01$	$.20 \pm .01$	$.41 \pm .09$	$.15 \pm .03$
german	700\20	$.51 \pm .09$	$.48 \pm .06$	$.49 \pm .04$	$.50 \pm .04$	$.49 \pm .05$	$.51 \pm .08$
waveform	400\21	$.23 \pm .09$	$.23 \pm .02$	$.22 \pm .01$	$.22 \pm .01$	$.23 \pm .01$	$.25 \pm .04$
cancer	192\9	$.56 \pm .09$	$.55 \pm .08$	$.58 \pm .06$	$.56 \pm .07$	$.56 \pm .07$	$.58 \pm .13$
flare solar	108\9	$.59 \pm .02$	$.60 \pm .02$	$.57 \pm .02$	$.57 \pm .02$	$.59 \pm .02$	$.58 \pm .02$
diabetes	468\8	$.48 \pm .04$	$.48 \pm .03$	$.50 \pm .02$	$.51 \pm .02$	$.47 \pm .02$	$.51 \pm .04$

The visualization of the Banana results is shown in figure 4.5. It is interesting to note from figure 4.5 that the variational classifier tends to place the inducing points at either side of the decision boundary. As the number of inducing points increases, the ‘resolution’ with which the inducing points define the decision boundary is improved.

4.5.3 Binary classification benchmark comparison

We compared the final hold out log probability for KLSp and GFITC on a variety of standard classification benchmarks. Using the same algorithmic setup described in section 4.5.2, we studied ten random train test partitions of the data. The results are shown in 4.3. The final behaviour of the two algorithms is broadly similar across the range of datasets.

4.5.4 Stochastic variational inference

The computational challenge of very large datasets can lead practitioners to choose statistical models that are simpler than they would otherwise choose or weaken the statistical motivation of their models. In this section we show that the stochastic minibatch version of our non-linear classifier, following from a principled Bayesian nonparametric derivation, can outperform a linear classifier on a real large dataset and that the training time is modest on a modern CPU. Our examples dataset consists of recorded features of all commercial flights in the United States between January 2008 and April 2008. There are roughly 5.8 million data points in this data set. Thus any algorithm that requires a full order N operation per iteration with several iterations is infeasible without significant distributed computing effort. The same

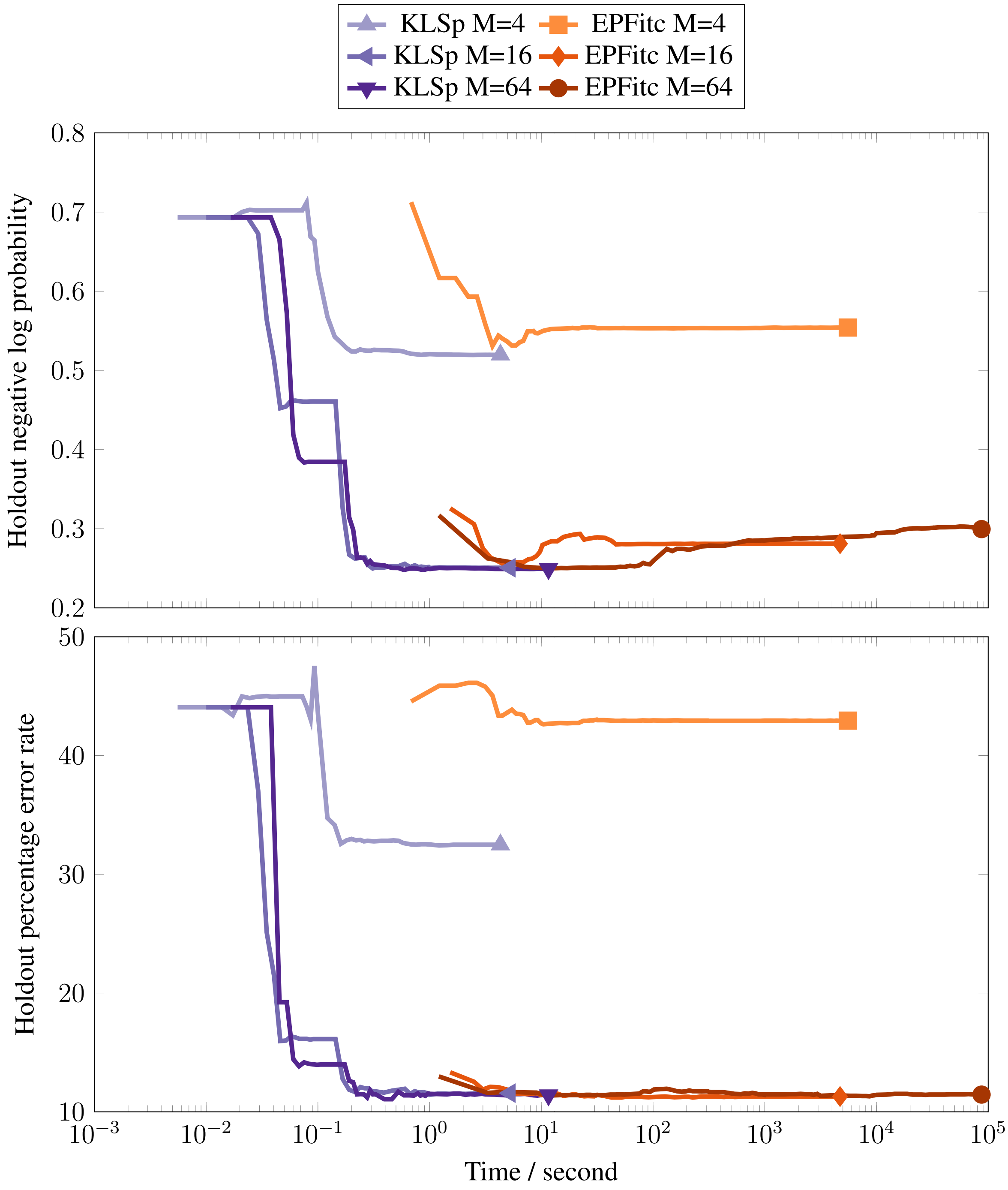

Fig. 4.3 Hold out negative log probabilities and error rates for the Banana dataset, using KLSp and GFITC, as a function of wall clock time. The effect of varying M, the number of inducing points, is also shown.

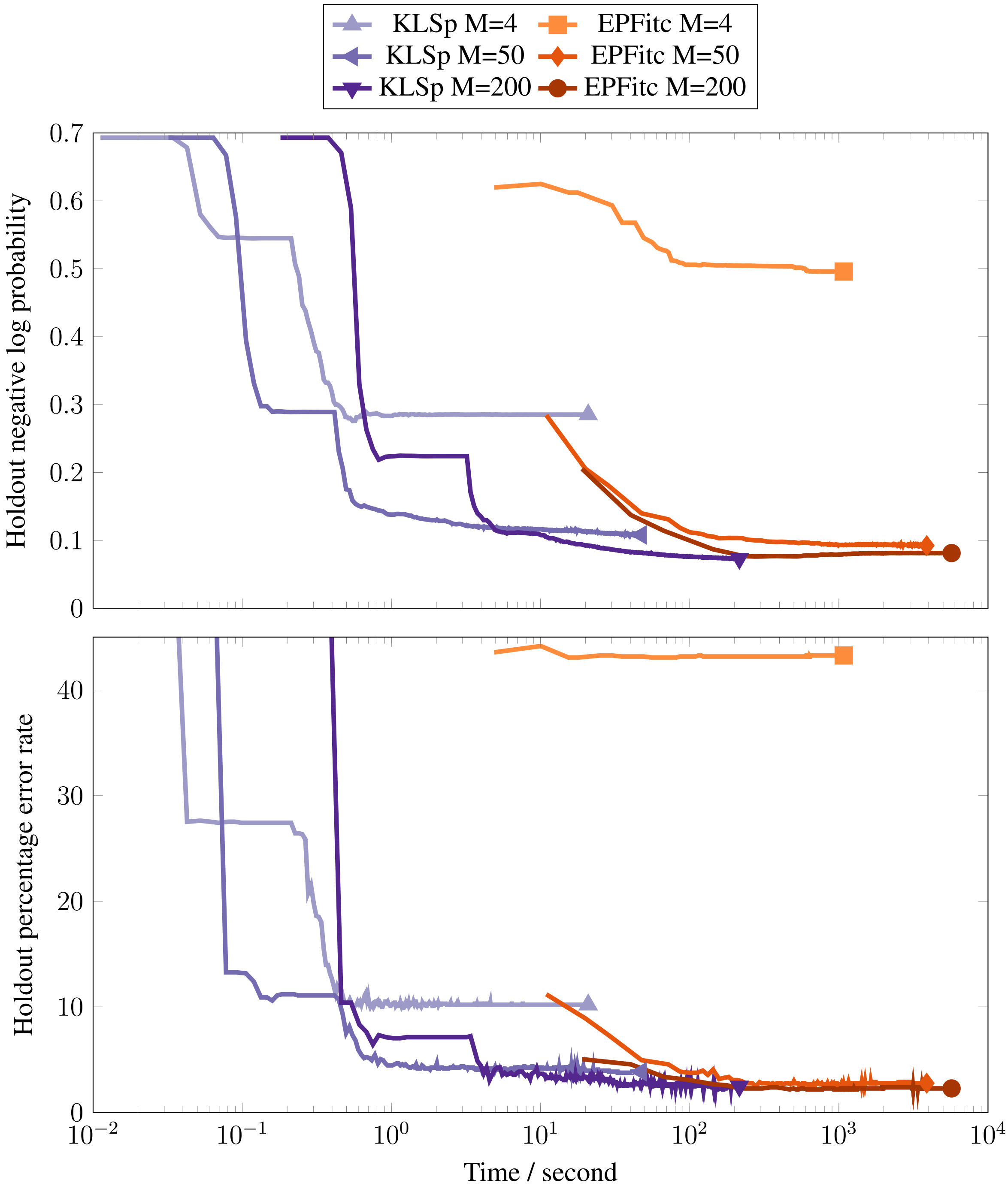

Fig. 4.4 Hold out negative log probabilities and error rates for the Image dataset, using KLSp and GFITC, as a function of wall clock time. The effect of varying M, the number of inducing points, is also shown.

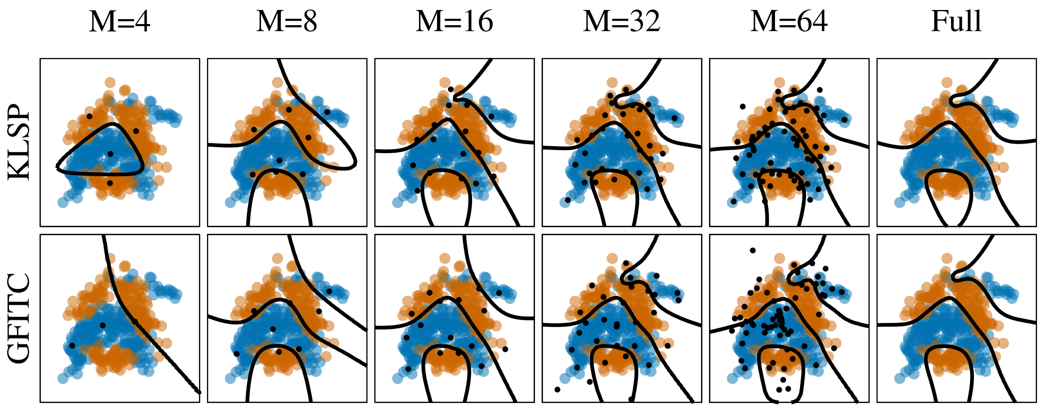

Fig. 4.5 The effect of increasing the number of inducing points for the Banana dataset. Each row corresponds to a different sparse classification method along with its non-sparse equivalent. The rows respectively correspond to the proposed variational KLSp method and the Generalized FITC method. The different columns correspond to different numbers of inducing points. For each plot, the orange and blue markers indicate the training data by class, the black lines indicate the decision boundary and the black markers indicate the learnt inducing input positions.

dataset was used by Hensman et al (2013) for their regression experiments. Recall that if we change the probit likelihood in our method for a Gaussian likelihood we recover the same bound as that method. The regression task in that paper was to predict the real valued average delay of the flight. Negative values denote that the flight was early. In this chapter we study an analogous classification problem which is to predict whether on not the flight was early or late. This data is produced by thresholding the real valued delay at 0. As in (Hensman et al., 2013) the features are the age of the aircraft (number of years since deployment), distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. As a baseline we used the LIBLINEAR (Fan et al., 2008) package which finds a linear classifier in less than a minute.

The Gaussian process model we used had zero mean function and the sum of a linear covariance function and a Matérn-3/2 covariance function. We used one length scale per input dimension in each case. This style of kernel is known as Automatic Relevance Determination and provides a convenient way of assessing the contribution of a feature. The parameters of the kernel were chosen by approximate type-II maximum likelihood, simultaneously with the variational approximation except we found that fixing the hyperparameters at the beginning of optimization helped reach a good solution. We used 150 inducing points. For stochastic optimization we used the AdaDelta method (Zeiler, 2012) as implemented in the library Climin (Bayer et al., 2016). We used a batch size of 50 and a step rate of 0.05 with a

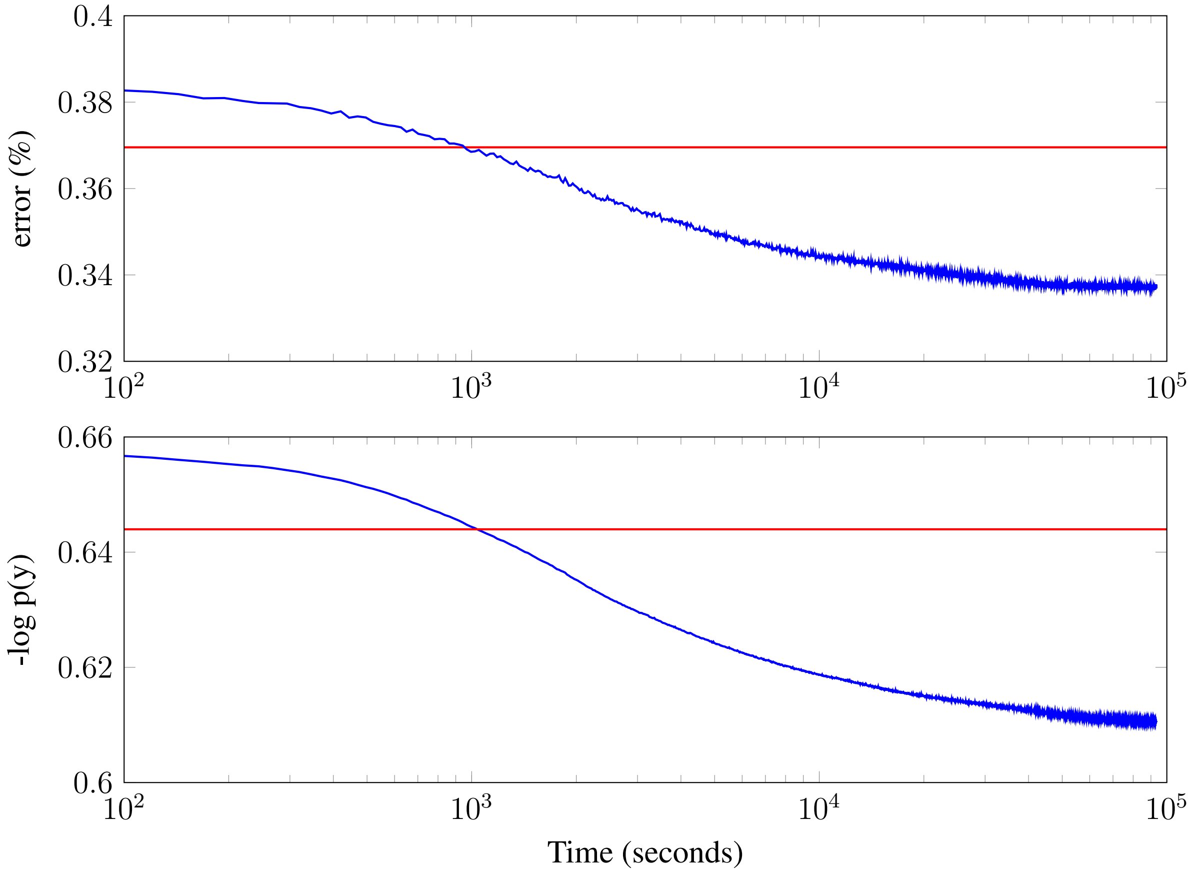

Fig. 4.6 Classification performance on the airline delay dataset as a function of time. The red horizontal line depicts performance of a linear classifier, whilst the blue line shows performance of the proposed stochastically optimized variational KLSp method.

momentum of 0.9. The inducing point positions were initialized using K-means clustering.

The results are shown in figure 4.6. We see that it took approximately 1000 seconds or roughly 17 minutes for the non-linear Gaussian process classifier to outperform the linear one. The learnt kernel had only small weight for the additive linear component suggesting that the data strongly favour a non-linear classifier. The most important kernel features were found to be time of day and time of year.

4.5.5 Multiclass classification experiments

We investigated multiclass classification on the MNIST dataset (LeCun et al., 1998), using the extension described in section 4.4. This dataset is heavily used in the field of machine learning. Whilst it is arguable that too much weight has been put on results on this dataset historically, the popularity of the data does have the benefit that it now enables comparison of many different classification algorithms on a high dimensional dataset. For this experiment we used the standard train/test partitions which consist of 60000 and 10000 points respectively. The data are 784 dimensional with each input data point consisting of the gray scale pixel values of the digits. There are 10 classes corresponding to the digits $0-9$ inclusive. We use 500 inducing points which are shared between the latent functions. Note that this gives 392000 variational parameters for the inducing point positions. There are $500 \times 10 = 5000$ variational parameters corresponding to the means of the latent functions at the inducing points and $125250 \times 10 = 1252500$ variational parameters corresponding to the function covariances. This is similar to the sort of challenging stochastic optimization problems that can be encountered with very large neural networks. We again used AdaDelta which has been found to be successful in this sort of challenging optimization problem. We used a minibatch size of 1000 and annealed the step size during optimization. The inducing point positions were initialized using K-means clustering. We used an RBF kernel plus a white noise term. We did not use ARD. It is generally accepted that the RBF kernel is a poor one for this sort of high dimensional problem because the Euclidean distances tend to become increasingly similar as the number of dimensions increase - an instance of the curse of dimensionality (Marimont and Shapiro, 1979). However for this example we wanted to take a very simple kernel to allow comparison with other kernel methods that have been applied to the same dataset.

Table 4.4 shows our result along with some other relevant methods. Unsurprisingly, the results for GPC with an isotropic RBF kernel do not give state of the art classification accuracy. At present, state of the art results on this dataset tend for the most part to be Deep Neural Network based and the best results are so low that in the subcommunity the dataset is no longer considered sufficiently challenging. By comparison there are very few results for Bayesian methods even published on this dataset and even fewer Bayesian nonparametric ones. Presumably this is due to the challenges of scaling probabilistic inference to datasets of this size. Given this paucity of results within the Bayesian subcommunity we would argue that any sensible result is of considerable scientific interest. One pre-existing result that does meet the criterion of being both Bayesian and nonparametric is that of Gal et al (2014) who give an accuracy of 5.95%. We would argue that this result is surprisingly inaccurate given that one would hope that the inductive bias of a Gaussian process latent variable model towards low dimensional manifolds would give it an advantage over Gaussian process classification. We would speculate that the approximate inference algorithm used in that paper is not well suited to the classification task. Another relevant comparison is to the Support Vector Machine (SVM) with the same kernel that we used. Given the simplicity of

Method	Reference	Permutation invariant?	Error rate
Linear classifier	(LeCun et al., 1998)	✓	12.0%
GP Latent variable model	(Gal et al., 2014)	✓	5.95%
One Nearest Neighbour	*Various*	✓	3.09%
GP Classifier RBF kernel	*This work*	✓	1.96%
SVM RBF kernel	*Various*	✓	1.4%
Committee of 7 convolutional networks	(Ciresan et al., 2011)	✗	0.27%

Table 4.4 A summary of some relevant classification method accuracies for the MNIST dataset. Many of the results can be found LeCun's website (LeCun, 2016).

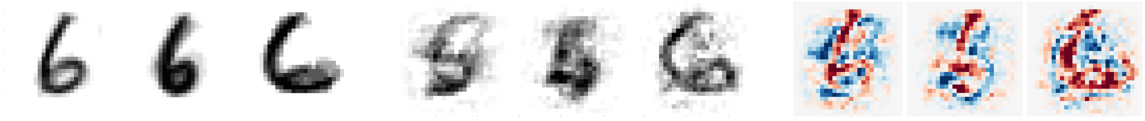

Fig. 4.7 Visualizing the inducing inputs for MNIST. The figure shows three groups of three inducing images. The left group shows three inducing inputs at their K-means initialization. The middle group shows the same inducing inputs after optimization. The right group shows the vector difference between the initial and final states.

the method and the strong theoretical grounding this result is impressive. We note that in general some potential benefits of GPC over an SVM are calibrated probabilistic predictions and the potential to scalably learn the parameters of a more complex covariance function.

Since the inducing points in our example lie in the input space and the input space in this case is a space of images, the inducing points can be themselves visualized usefully as images. Figure 4.7 shows some examples of what happens to the inducing points during optimization. They are initialized using the K-means algorithm and as such they will start out looking like digits, as for example, in the left group where we have three sixes. However the inducing points move away from their initialization towards the classification decision boundary. The middle group shows that the same inducing points in the first two cases look like 'hybrids' of different digits; respectively a 6/5 hybrid and a 6/4 hybrid. Recall that very similar decision boundary behaviour was observed in the simple example of section 4.5.2. The third inducing point looks like a rarer example from the input distribution of sixes.

4.6 Conclusion

There are two crucial challenges that often emerge in Gaussian process inference: namely the challenge of approximation required for working with non-Gaussian likelihoods and the challenge of scaling these methods to large datasets. In this chapter we have introduced a method to address these challenges simultaneously, concentrating on the important special case of classification. We argued that the method should use sparsity, present an integrated variational approach, leverage continuous inducing point optimization and have a scalable and convergent extension using minibatches (section 4.2). We then contributed a method that had all these features and studied it empirically. Without Stochastic Variational Inference we showed that the method has competitive performance with the popular Generalized FITC algorithm (section 4.5.2). Using SVI we are able to give good quality results on important large datasets like the MNIST dataset for the first time (section 4.5.5). In the next chapter we will see another example non-Gaussian likelihood, namely the Poisson likelihood, and investigate a method that is able to relax the strong Gaussian assumption on $q(f_Z)$ by using an MCMC/variational hybrid algorithm.

Chapter 5

MCMC for Variationally Sparse Gaussian Processes

5.1 Introduction

In chapter 4 we considered the application of a Gaussian approximation for the variational distribution $q(f_Z)$ over inducing outputs to the task of scalable inference with non-Gaussian likelihoods. Using the language introduced in section 2.2 we made a fixed form assumption. Whilst a free form assumption could give a smaller $\mathcal{KL}$-divergence, it only has a closed form solution in the case of a Gaussian likelihood. In this chapter we investigate a method of sidestepping this obstacle by using Markov Chain Monte Carlo (MCMC) to draw samples from the variationally optimal $q(f_Z)$ distribution. This may initially seem like an odd direction to go in because one of the traditional benefits of variational methods was that they provided fast deterministic alternatives to sampling methods. A key theme of this thesis, however, is that the variational framework can be used to make principled sparse approximations to Gaussian process posteriors, reducing $\mathcal{O}(N^3)$ computations to $\mathcal{O}(NM^2 + M^3)$. As we shall see, this benefit can be achieved independently of the strong distributional assumptions of the previous chapter. In short, we aim for a 'hybrid' approximation that inherits a principled approach to sparsity from variational methods and improved distributional flexibility from MCMC.

This improved flexibility allows us to increase the complexity of the models used, in particular by giving a fully Bayesian treatment of the kernel and likelihood hyperparameters (section 5.3). This means our approximation target is the full Bayesian posterior rather than the type-II maximum likelihood solution, the former being more philosophically pure from a Bayesian perspective. From a pragmatic modelling perspective, it is clearly sometimes beneficial to be able to coherently incorporate domain knowledge about process hyperparameters.

There is a risk that the mathematics around these generalizations can become complex and opaque. We highlight the concept of *variational potentials* (section 5.2) as a method for understanding sparse variational approximations. This provides us with interpretable analogies to familiar graphical model concepts, which have in turn guided this research.

The specific realization of MCMC we investigate (section 5.4) is Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2010) across both hyperparameters and whitened latent function values. The full approximation procedure is detailed in section 5.5.

Empirically (section 5.6), by studying a variety of examples, we find some important cases where the Gaussian approximation works well relative to the more general free form approximation. In these situations, relaxing some of the strongest assumptions of the simpler approximation has allowed us to gain confidence in their efficacy. In other cases, we find that the approximation of this chapter successfully models qualitative aspects of the posterior that would be difficult or impossible with the Gaussian assumption and a point estimate of the hyperparameters.

5.2 Variational potentials

We return to our definition of the $\mathcal{KL}$-divergence between processes which was derived in 3.3.3 and which was further discussed in section 4.3. We repeat it here for ease of exposition

$$\mathcal{K} = \mathcal{KL}[q(f_Z)||p(f_Z)] - \sum_{y,d \in Y,D} \mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] + \log p(Y)\,. \tag{5.1}$$

For conciseness, we have suppressed some dependencies on the inducing points Z in the likelihood terms. Let us have a closer look at the likelihood terms:

$$\mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] = \int_{\mathbb{R}^M} \int_{\mathbb{R}} p(f_d|f_Z) q(f_Z) \log p(y|f_d) df_d df_Z\,. \tag{5.2}$$

With this in mind, we define a *variational potential*[1] $\phi(y|d, f_Z)$ in terms of its logarithm

$$\log \phi(y|d, f_Z) := \int_{\mathbb{R}} p(f_d|f_Z) \log p(y|f_d) df_d\,. \tag{5.3}$$

Using this new definition we can rearrange equation (5.1) as follows:

[1] We avoid the term variational likelihood because the variational potentials do not necessarily normalize as a function of the data point y.

$$\mathcal{K} = \mathcal{KL}\left[q(f_Z)||\frac{p(f_Z)\prod_{y,d\in Y,D}\phi(y|d,f_Z)}{C(Y)}\right] - \log C(Y) + \log p(Y) \tag{5.4}$$

where

$$C(Y) = \int_{\mathbb{R}^M} p(f_Z)\prod_{y\in Y}\phi(y|d,f_Z)df_Z\,. \tag{5.5}$$

To prove this relation we manipulate equation (5.1)

$$\mathcal{K} = \mathcal{KL}[q(f_Z)||p(f_Z)] - \sum_{y,d\in Y,D}\mathbb{E}_{q(f_d)}\left[\log p(y|f_d)\right] + \log p(Y) \tag{5.6}$$

$$= \int q(f_Z)\log\frac{q(f_Z)}{p(f_Z)}df_Z - \sum_{y,d\in Y,D}\left[\int q(f_Z)\log\phi(y|d,f_Z)df_Z\right] + \log p(Y) \tag{5.7}$$

$$= \int q(f_Z)\log\frac{q(f_Z)\,C(Y)}{p(f_Z)\prod_{y,d\in Y,D}\phi(y|d,f_Z)}df_Z - \log C(Y) + \log p(Y)\,. \tag{5.8}$$

The result then follows. Note that the variational potentials, unlike the original likelihood, do not factorize across data points. Also note that we pulled the constant $C(Y)$ into the ratio of logarithms in order to ensure that the right hand entry of the $\mathcal{KL}$-divergence in equation (5.4) was a normalized probability distribution in f_Z. None of the terms outside the $\mathcal{KL}$-divergence depend on $q(f_Z)$, so for fixed Z we need only consider this $\mathcal{KL}$-divergence term. $\mathcal{KL}$-divergence reaches its minimal value of zero when the two input probability distributions are equal. Therefore we can write down the optimal form for $q(f_Z)$ by inspection of equation (5.4)

$$q^*(f_Z) = \frac{p(f_Z)\prod_{y,d\in Y,D}\phi(y|d,f_Z)}{C(Y)}\,. \tag{5.9}$$

Thus, similarly to the case of free form factorized approximations considered in section 2.2.2, we can completely sidestep the need to use variational calculus and gain more insight into the result.

In the case of regression with a Gaussian likelihood and noise variance σ^2, each variational potential is an unnormalized Gaussian function. The inducing output prior $p(f_Z)$ is also a Gaussian distribution. As a product of Gaussian terms which must normalize, the optimal distribution in this case must necessarily be a Gaussian distribution. The precise form was derived by Titsias (2009a)

$$q^{*}_{\text{Gauss}}(f_Z) = \mathcal{N}(f_Z|m_{\text{Gauss}}, \Sigma_{\text{Gauss}})\,. \tag{5.10}$$

with

$$m_{\text{Gauss}} = \sigma^{-2}K_{Z,Z}\Delta K_{Z,D}y_D \tag{5.11}$$

$$\Sigma_{\text{Gauss}} = K_{Z,Z}\Delta K_{Z,Z} \tag{5.12}$$

$$\Delta = (K_{Z,Z} + \sigma^{-2}K_{Z,D}K_{D,Z})^{-1} \tag{5.13}$$

Note that whilst the interpretation in terms of variational potentials is important for understanding optimization of the $\mathcal{KL}$-divergence, the analogy breaks down somewhat when we consider prediction. That is to say that the prediction points (y^{*}, x^{*}) are not computed using a variational potential. Rather, they come from the definition of the approximating process in equation (4.19) of section 4.3.

Returning to the general case, if we substitute the optimal variational distribution for the inducing points we get a collapsed variational bound

$$\mathcal{K}^{*} = \log p(Y) - \log C(Y) \tag{5.14}$$

Therefore the remaining $\mathcal{KL}$-divergence between the processes has the elegant interpretation of being the difference between the marginal likelihood of the true model and the normalizing constant associated with the variational potentials. The true marginal likelihood is not a function of the inducing point positions and can be ignored as an additive constant in inducing point selection. In the case where (with no computational gain) we set the inducing points Z to equal the data points D then the variational potentials are equal to the real likelihoods and $C(Y) = p(Y)$ giving a $\mathcal{KL}$-divergence of 0. This must be true since this case corresponds to exact inference.

We can contrast this variational potential view to the Quinoñero-Candela and Rasmussen (2005) view of sparse Gaussian processes, which involves prior approximation rather than likelihood approximation. The latter has the benefit that we know additional computation in the form of additional inducing points will bring us closer to the true posterior, emerging as it has as a logical consequence of the variational framework.

Titsias showed (Titsias, 2009a) using the augmentation argument that for the case of a Gaussian likelihood with noise variance σ^2 the $\mathcal{KL}$ reduces to

$$\mathcal{K}^{*}_{\text{Gauss}} = \log p(Y) - \log \mathcal{N}(y_D|\sigma^2 I + K_{D,Z}K^{-1}_{Z,Z}K_{Z,D}) + \frac{1}{2\sigma^2}\operatorname{Tr}\left(K_{Z,Z} - K_{D,Z}K^{-1}_{Z,Z}K_{Z,D}\right) \tag{5.15}$$

As Titsias noted, the second term is precisely the marginal likelihood of the DTC covariance approximation (see also section 4.2.2) and the third 'trace term' differentiates this approximation from the DTC approximation.

In the rest of this chapter we will be interested in the non-conjugate likelihood case. Whilst there is not an analytic $q(f_Z)$ in this case, we can still utilize the insight that comes from the variational potential view of the sparse variational approximation. But before we further discuss non-conjugate inference, we discuss the extension of the variational framework to fully Bayesian models.

5.3 A Bayesian treatment of the hyperparameters

Consider the extension of the model in equation (4.1) to include a prior on hyperparameters.

$$\theta \sim p(\theta) \tag{5.16}$$

$$f \sim \mathcal{GP}(0, K(\theta)) \tag{5.17}$$

$$y \,|\, d, f \sim h(f(d)), \qquad y, d \in Y, D \text{ i.i.d} \tag{5.18}$$

The approximating process in this case is taken to have the following finite dimensional marginals

$$q(f_*, f_D, f_Z, \theta) = p(f_*|, f_D, f_Z, \theta) p(f_D|f_Z, \theta) q(f_Z, \theta) \tag{5.19}$$

Contrasting this to the approximating distribution given in equation (4.19) we see that we now maintain a joint variational distribution over the inducing function values and the hyperparameters rather than the inducing function values alone. We do not assume that $q(f_Z, \theta)$ factorizes.

The $\mathcal{KL}$-divergence from the approximating process to the posterior process in this case is given by

$$\mathcal{K} = \mathcal{KL}[q(f_Z, \theta)||p(f_Z, \theta)] - \sum_{y,d \in Y,D} \mathbb{E}_{q(f_d)}[\log p(y|f_d)] + \log p(Y) \tag{5.20}$$

Where $q(f_d)$ is now the corresponding marginal distribution of the process defined by (5.19). The variational potential $\phi(y|d, f_Z, \theta)$ in this case is given by

$$\log \phi(y|d, f_Z, \theta) = \int_{\mathbb{R}} p(f_d|f_Z, \theta) \log p(y|f_d) df_d \tag{5.21}$$

where we have reintroduced the dependence on θ. Following a similar argument to before the $\mathcal{KL}$-divergence between processes can be rearranged to give

$$\mathcal{K} = \mathcal{KL}\left[q(f_Z, \theta)||\frac{p(f_Z, \theta)\prod_{y,d\in Y,D}\phi(y|d, f_Z, \theta)}{C(Y)}\right] - \log C(Y) + \log p(Y) \tag{5.22}$$

where the new definition of the normalizing constant is

$$C(Y) = \int p(f_Z, \theta)\prod_{y,d\in Y,D}\phi(y|d, f_Z, \theta) df_Z d\theta\,. \tag{5.23}$$

The collapsed variational bound has the same form as in equation (5.14), substituting this new normalizing constant in place of the previous definition. By extending our model to a fully Bayesian treatment we have introduced further intractability into the variational objective (5.4) which is already challenging for non-Gaussian likelihoods. In the next section we detail an approximation method capable of tackling even this harder case.

5.4 Sparse variational Markov chain Monte Carlo

We need to consider the fact that relatively few likelihoods result in a tractable objective for equation (5.4) and even fewer have a tractable objective for the Bayesian extension in equation (5.22). In chapter 4 we considered restricting the variational family for the inducing points $\mathcal{Q}_{\text{ind}}$ to be a family of Gaussians which is tractable when a point estimate of the hyperparameters is used. This family will not however usually contain the optimal variational distribution. In this chapter we will consider taking the analogy with an approximating model seriously and drawing samples from the optimal variational distribution for f_Z, using Markov Chain Monte Carlo methods (MCMC) (Metropolis et al., 1953), concentrating in particular on Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2010). An idea in a similar spirit was suggested in (Titsias et al., 2011) but was dismissed before implementation because ‘prediction in sparse GP models typically involves some additional approximations’. Our interpretation in terms of a $\mathcal{KL}$-divergence between the approximating and posterior processes clarifies that no such further approximation is required.

To use MCMC, we only need to know the desired probability distribution up to a multiplicative constant. That is to say we do not need to know the constant $C(Y)$ which in the non-conjugate case relies on a difficult integral. If we are intending on doing MCMC anyway why consider a variational approximation? By making the assumption that the approximating distribution has the form in equation (4.19) or equation (5.19), we find that evaluating the necessary function for MCMC (and derivatives in the case of HMC) requires $\mathcal{O}(NM^2 + M^3)$ rather than the $\mathcal{O}(N^3)$ required for MCMC in the original model. Thus our proposed method makes a minimal variational assumption to render the approximation tractable but inherits many of the advantages of MCMC. Note that the method we suggest is not equivalent to performing MCMC with a sparse prior covariance and a non-Gaussian likelihood. Instead we are sampling from the variational distribution in the approximating family that is optimal in the sense of minimizing $\mathcal{KL}$-divergence to the posterior.

The variational potentials mean that the MCMC problem looks very much like that of a hierarchical latent Gaussian model albeit with a non-factorizing set of data potential terms. MCMC in hierarchical latent Gaussian models is a challenging problem because typically there is a strong correlation between the values of the hyperparameters and the latent Gaussian values themselves (Filippone et al., 2013; Murray et al., 2010). Here, motivated by the 'whitening' reparameterization of (Murray et al., 2010) we propose a method that is able to update the whitened variables and the hyperparameters at the same time. We proceed to explain in more detail. Consider the log density $H(f_Z, \theta)$ of the unnormalized target distribution in the fully Bayesian case

$$H(f_Z, \theta) = \log p(\theta) + \log p(f_Z|\theta) + \sum_{y,d \in Y,D} \int_{\mathbb{R}} p(f_d|f_Z, \theta) \log p(y|f_d) df_d \,. \tag{5.24}$$

Here we have substituted the definition of the variational potentials. Let Λ be the Cholesky of the prior covariance $K_{Z,Z}$ of the inducing function values f_Z so that $\Lambda\Lambda^T = K_{Z,Z}$ and Λ is lower triangular. We consider the transformation to whitened variables v defined by

$$v = \Lambda^{-1}(\theta) f_Z \tag{5.25}$$

These variables are *apriori* Gaussian with mean zero and isotropic unit variance noise, hence the 'whitened' name. If we were attempting to draw samples from the prior this would reduce correlation and ill-conditioning which would improve mixing. In the case where the likelihood does not strongly constrain the latent function then presumably this posterior will not be too different from the prior and the whitened representation will still improve mixing. In fact, even when the likelihood does more strongly constrain the function values we still often observe an improvement in mixing using the whitened representation (5.25), relative to the unwhitened representation.

The Jacobian of the change of variables (5.25) is $\det \Lambda^{-1}(\theta)$ as discussed in Appendix E. The unnormalized log density in the whitened representation is given by:

$$H(v,\theta) = \log p(\theta) + \log \mathcal{N}(v,0,I) + \sum_{y,d \in Y,D} \int_{\mathbb{R}} p(f_d|\Lambda(\theta)v,\theta) \log p(y|f_d) df_d \,. \tag{5.26}$$

Since we are interested in Hamiltonian Monte Carlo for the log density defined in equation 5.26, we will need to compute the derivatives of $H(v,\theta)$ with respect to the whitened variables v and the covariance hyperparameters θ. The target $H(v,\theta)$ is defined in terms of the Cholesky matrix Λ. Therefore to implement the derivatives, we use reverse mode differentiation of the Cholesky decomposition, as first detailed by Smith (1995) and discussed further in section 6.5.2.

We also consider how to choose the inducing inputs with this approximation. Since these are variational parameters we will want to minimize the variational objective with respect to them rather than sample them. Define $h(f_Z,\theta)$ such that

$$h(f_Z,\theta) := \exp\left\{H(f_Z,\theta)\right\} = p(f_Z,\theta) \prod_{y,d \in Y,D} \phi(y|d,f_Z,\theta) \tag{5.27}$$

is the unnormalized version of the optimal variational distribution $q^*(f_Z,\theta)$ and has $C(Y)$ as its normalizing constant. Taking derivatives of equation (5.14) with respect to Z gives us

$$\begin{aligned}
\frac{\partial \mathcal{K}^*}{\partial Z} &= \frac{\partial}{\partial Z}\left\{-\log C(Y)\right\} && (5.28)\\
&= -\frac{1}{C(Y)} \frac{\partial}{\partial Z} \int h(f_Z,\theta) df_Z d\theta && (5.29)\\
&= -\frac{1}{C(Y)} \int h(f_Z,\theta) \frac{\partial}{\partial Z}\Big[\log h(f_Z,\theta)\Big] df_Z d\theta && (5.30)\\
&= -\mathbb{E}_{q^*(f_Z,\theta)}\left[\frac{\partial H(f_Z,\theta)}{\partial Z}\right]\,. && (5.31)
\end{aligned}$$

Since drawing samples from $q^*(f_Z,\theta)$ is our goal with Hamiltonian Monte Carlo, evaluating a Monte Carlo estimate of this gradient seems possible but taxing. Instead in the next section we detail a pragmatic approximation scheme that leverages the Gaussian approximation.

5.5 A recipe for using variational MCMC in practice

Having described the theory behind our approximation strategy we now outline the practical algorithmic steps we take to obtain our approximate posterior.

1. We first use the Gaussian approximation described in chapter 4. In more detail, we choose a set of optimal parameters by maximizing the evidence lower bound (4.22), substituting the desired likelihood. The optimization parameters are the model hyperparameters θ, the inducing point positions Z, and the parameters of the assumed Gaussian distribution $q(f_Z) = \mathcal{N}(f_Z|m, \Sigma)$ on the inducing outputs. Of the two Gaussian distribution parameters, the variational mean m is used directly, whereas the covariance Σ is specified using a lower triangular matrix L such that $LL^T = \Sigma$. The inducing inputs Z are initialized using K-means clustering. For smaller datasets, we use the L-BFGS-B optimization method (Zhu et al., 1997) on the variational parameters and the covariance parameters. For larger datasets, we use stochastic gradients as described in chapter 4, specifically using the AdaDelta optimizer (Zeiler, 2012). We fix the inducing point positions Z early in optimization because this often results in a better optimum. The optimal values for θ, Z, m and L are then used in the next stage of the approximation process.

2. We use the result of the Gaussian approximation to the posterior to initialize the sampling approximation to the posterior. We do not try to learn the optimal inducing point positions for the free form variational distribution. Instead, we fix the inducing point positions Z to equal the optimal Gaussian ones from the previous algorithmic step. We initialize the model hyperparameters θ to the approximate maximum aposteriori (MAP) values obtained from the Gaussian approximation. The whitened inducing outputs v, defined in equation (5.25), are initialized by exploiting the Gaussian inducing output distribution $q(f_Z)$ used in the previous algorithmic stage. In particular, we draw a sample for f_Z from the Gaussian distribution and then transform it to the whitened space using the relation in equation (5.25).

3. We next tune the sampling parameters of the HMC algorithm using Bayesian optimization. The sampling parameters that need tuning are the number of leapfrog steps and the step length. To do this, we broadly use the approach of (Wang et al., 2013), by using Bayesian optimization on the expected squared jump distance, penalized by the square root of the maximal number of leap frog steps. We however do not follow their adaptive prescription, instead simply using 30 different runs of 30 iterations each.

4. Finally, we run the tuned HMC to obtain predictions. In more detail, we apply the joint HMC transition operator to the model hyperparameters θ and the whitened inducing outputs v with the log density given in equation (5.26). We then transform the samples of the whitened variable v to samples of f_Z using the relation in equation (5.25). Taking

the samples for θ and f_Z obtained, we make predictions by replacing the expression for $q(f_Z, \theta)$ in equation (5.19) with the corresponding Monte Carlo average.

5.6 Experiments

5.6.1 Binary classification

We again use the Image dataset which is a binary classification benchmark (Rätsch et al., 2001). The input data is 18 dimensional. We use an RBF kernel with one automatic relevance determination parameter per input dimension. The kernel lengthscales and variance are given a $Gamma(1, 1)$ prior. The data is partitioned into 10 random train/test splits with 1000 and 1019 members respectively. We follow the general recipe for inference described in section 5.5 with some modifications which we now describe.

We first investigate the efficacy of variational inducing point optimization. To do this, we compare the full procedure in section 5.5 to doing the same procedure, but with the inducing points clamped throughout at their K-means initialization. The hold out log predictive densities are shown in the left hand panel of figure 5.1. For lower inducing point numbers, there is a significant gain to be found from using the variational criterion to find the inducing point positions. It is not surprising that the absolute gains are smaller for a large number of inducing points, though these gains might still be important if a high predictive accuracy was required.

Next we compared the full MCMC approximation to the Gaussian approximation step only including the variational inducing point optimization. This gives a measure of how much the MCMC approximation improves the hold out log predictive density for this dataset. The results are shown in the right hand panel of figure 5.1. It can be seen for small numbers of inducing points there is negligible improvement. This suggests that, in this case, uncertainty in the kernel hyperparameters and non-Gaussianity in the posterior over the inducing outputs are not the dominant source of approximation error. For larger numbers of inducing points there is a consistent improvement over the Gaussian approximation suggesting the qualitative factors introduced in the MCMC approximation do play some role in this case. However the improvement is small in magnitude when compared to the absolute effect of varying the number of inducing points that can be observed in the left hand panel of 5.1. This suggests that for practical purposes the Gaussian approximation works relatively well for this usage case, particularly given that it gives an analytic lower bound on the model evidence and works faster. Relaxing some of the strong assumptions of the Gaussian method of chapter 4 in this manner allows us to gain confidence in the quality of the approximation.

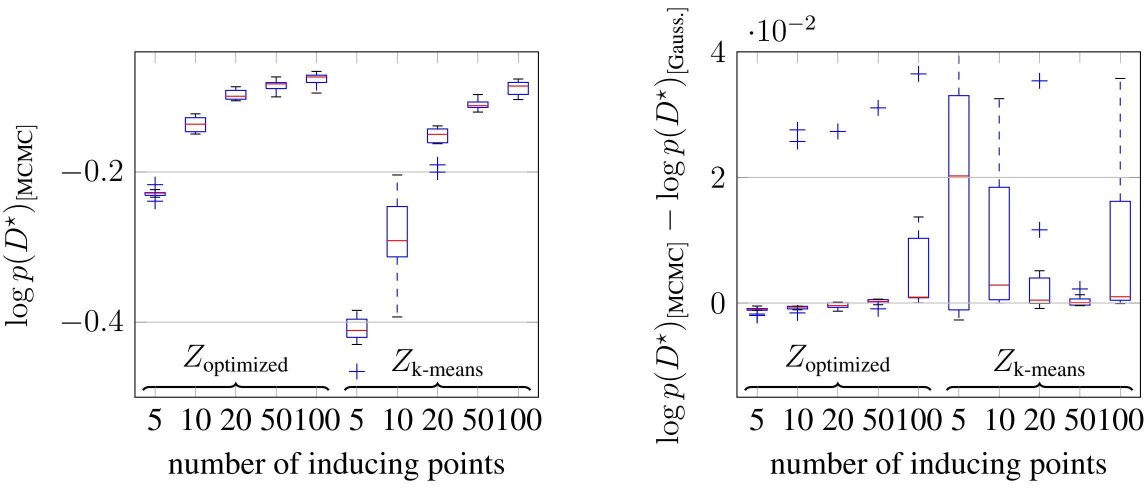

Fig. 5.1 Testing the variational MCMC method on the *Image* dataset (Rätsch et al., 2001). Left: the difference in hold out log predictive density for differing numbers of inducing points for the variational MCMC method with and without variational optimization of the inducing points under stage 1 of the approximation recipe given in section 5.5. Right: the difference in hold out log predictive density comparing variational MCMC to using the Gaussian approximation only.

5.6.2 Multiclass Classification

In section 4.4 we discussed multiclass classification using a Gaussian approximation. Additionally we made an assumption that the variational distribution factorizes across the latent functions corresponding to each class. This had two main motivations. First, the factorization meant that the robust max likelihood quadrature was tractable using a single one dimensional numerical quadrature. Second, the memory requirements of representing a covariance that does not factorize across the different functions grows very quickly. Variational MCMC thus has some potential advantages here. The necessary likelihood quadratures are similar to the factorized Gaussian case but do not rely on the strong factorizing assumption. The memory usage may also scale better given that the covariance is not explicitly represented. Instead we store samples which can be cheaper[2]. Thus it is of interest to compare the variational MCMC approach to the factorized Gaussian approach in this instance empirically.

First we compare the two methods on a synthetic dataset to allow visualization in a controlled simple situation as shown in figure 5.2. We can see that on the whole the variational MCMC method gives a more conservative estimate of the predictive probabilities.

[2] It is worth noting that this might not be the only way to address the memory scaling issue. In the case where the optimal variational distribution has a low rank we may be able to get away with using fewer variational parameters but we have not yet investigated this direction.

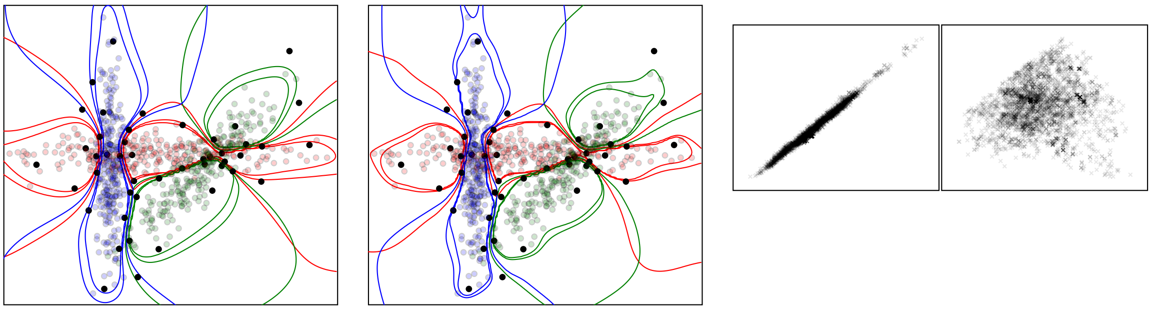

Fig. 5.2 Comparison of factorized Gaussian and variational MCMC for a synthetic multiclass classification problem. Left: The factorized Gaussian approximation. The coloured points show data points for each class. The coloured lines show posterior probability contours for each class. The contours are chosen to be at [0.3,0.95,0.99]. The black points show the location of the inducing points. Middle: The same plot but using free form variational MCMC. Right: Some examples of posterior pairwise correlations of latent functions for different classes but at a fixed point. The posterior shows correlations and non-Gaussianity that would not be captured by a factorized Gaussian approximation.

This could make a significant difference in applications where the loss associated with an overconfident prediction is high.

We also compared the multiclass methods on the MNIST example given in section 4.4 which has significantly more input dimensions and data points then the synthetic example. The variational MCMC example was very slow in this case because the Metropolis correction involves a full linear parse over 60000 data points. The classification accuracy did not improve in this case. The hold out log predictive probability gave a marginal improvement from -0.068 to -0.064. Again we would conclude that for this application if these are the metrics of interest then the factorized Gaussian approximation gives similar performance with a lower compute time. The memory cost of the different possible approximations to the inducing output distribution is shown in table 5.1. Since in this case the factorized Gaussian approximation gives a similar performance to the sampling approximation it also represents a good choice in terms of memory usage.

5.6.3 Log Gaussian Cox processes

We investigate the application of our sparse MCMC approach to inference in point process models from spatial statistics. The general area of spatial statistics is one where the qualitative difference of having reasonable uncertainty estimates for the covariance hyperparameters, as opposed to solely a point estimate, is often relatively easy to motivate. Often in spatial statistics one is interested in interpreting these hyperparameters. For instance, one might wish to decide whether a given dataset shows evidence of spatial correlation and what

Factorized Gaussian	Full Gaussian (not implemented)	Sampling
1.3	13	1.5

Table 5.1 MNIST memory cost in millions (2 s.f) of real numbers for representing the distribution of the inducing outputs. We use 500 inducing points. The sampling approximation used 300 samples. The Full Gaussian approximation would mean representing a covariance across all ten latent functions. Note that the sampling approximation uses the factorized approximation as an intermediate step. All of the methods need to store the inducing inputs which are shared across latent functions and require storage of 0.39 million real numbers.

characteristics that correlation has, rather than only being interested in the predictive intensity, as is often the case in machine learning applications. Since spatial statistical models are often computationally demanding any solution needs also to be scalable.

In this section we study a discretized version of the Log Gaussian Cox process (Møller et al., 1998). As discussed in section 3.6.2 the general form for any Gaussian process based Cox process is

$$\begin{aligned} f &\sim \mathcal{GP}(m, K) \\ h &= \rho(f) \\ Y|h &\sim \mathcal{PP}(h)\,. \end{aligned} \tag{5.32}$$

The notation is the same as section 3.6.2. For the purposes of this section we assume that the inverse link function ρ is the exponential function so that:

$$h = \exp(f) \tag{5.33}$$

It is this additional assumption that characterizes the Log Gaussian Cox process (LGCP) of Møller et al (1998). Unfortunately, the likelihood involves an integral (3.42) over the whole latent function, which tends to add complications to approximate inference. Lloyd et al (2015) study sparse variational inference for another Gaussian process based Cox process, specifically one with a squared inverse link function, so that $h = f^2$. Under this assumption and a restricted class of kernel functions, the variational inference is tractable even with the difficult integral of the function term. Here we adopt an approach that is able to tolerate a variety of inverse link functions, although the likelihood quadrature is particularly convenient for the LGCP case since it can be done analytically. The approach makes relatively few assumptions on the choice of covariance function, which allows this to be a modelling choice.

For this section we introduce a ‘binning’ on the space. To each bin there then corresponds a Poisson likelihood rather than a Poisson process we had for the unbinned space. The Poisson rate for a given bin is equal to the hypervolume of that bin multiplied by the function value at the center of the bin. This is a common assumption in applied work and as such provides a good comparison point for investigating the quality of variational MCMC approximations. Intuitively at least, for smooth covariance functions the approximation recovers the full Cox process as the bins become small. We consider the effect of different bin sizes in some of the experiments that follow.

We studied the quality of the approximation on the one dimensional coal mining disaster data. We chose an RBF kernel and placed Gamma priors on the variance and lengthscale. Three methods were considered:

1. Gold standard exact MCMC using the non-sparse analogue of our Hamiltonian Monte Carlo algorithm.

2. The Gaussian approximation with a MAP estimate of the covariance hyperparameters.

3. Our variational MCMC approximation.

For both sparse approximations we used 30 evenly spaced inducing points which we kept fixed rather than optimizing their position.

The results are shown in figure 5.3. The approximate posterior over intensities between the full MCMC and the variational MCMC in the left panel shows very close agreement. The Gaussian approximation by comparison exhibits clear distortion. The kernel hyperparameter estimates from the variational sampling algorithm in the right panel show good agreement with the full MCMC in general, except that for both the variance and the lengthscale there is a slight overestimate of the upper tail.

We investigated the use of variational MCMC on the Pine sapling data (Møller et al., 1998), shown in the top left panel of figure 5.4. We compared against the full MCMC method and also used this opportunity to investigate the effect of bin size. For the sparse methods, we used 225 inducing points, which we initialized in a grid and then variationally optimized during the Gaussian approximation phase. We investigated the methods for a 32×32 grid and a 64×64 grid. To give a measure of the computational demands of each method we computed the time for both the variational MCMC and exact MCMC methods to obtain one effective MCMC sample.

The results are a shown in figure 5.4. On the 32×32 bind grid it can be seen that the variational MCMC agrees well with the exact MCMC. The variational MCMC was considerably faster taking 3.4 seconds to obtain an effective sample as opposed to 554

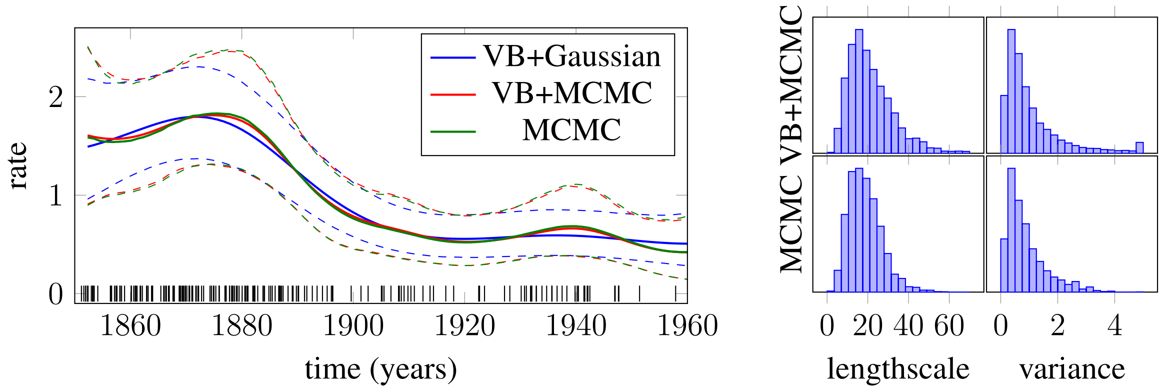

Fig. 5.3 Comparison of exact MCMC, variational MCMC, and the Gaussian variational approximation on the coal mining disasters dataset. Left: Approximate posterior rates of the three methods. Vertical bars show data points. Right: A comparison of the posterior samples for the kernel hyperparameters generated by the two MCMC methods. The MAP estimate for the Gaussian approximation was (12.06,0.55).

seconds for the exact method. It can be seen that the 64×64 grid variational MCMC provides a much higher resolution approximation to the posterior intensity. On this finer grid, the variational method took 4.7 seconds per effective sample. By comparison the exact MCMC was prohibitively slow on this data because the linear algebra requires $\mathcal{O}(N^3)$ operations with $N = 4096$.

5.7 Conclusion

In the previous chapter we studied a fixed form, Gaussian, variational approximation to $q(f_Z)$ with a point estimate of the hyperparameters. In this chapter we studied the generalization to a Bayesian treatment of the hyperparameters with a free form joint variational distribution $q(f_Z, \theta)$. In both cases the variational approximations where extended to the other unknown variables using prior conditional matching (section 2.2.4). By using MCMC to draw samples from $q(f_Z, \theta)$, we were able to circumvent the lack of a tractable closed form for the optimal free form distribution. Thus we proposed an algorithm that was a variational/MCMC 'hybrid' that inherited a reduction in training complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2 + M^3)$ from the variational sparsity and a more flexible approximating distribution from MCMC. We highlighted the concept of variational potentials as a means to map these problems onto more familiar graphical model inference problems. A practical approximation scheme that used Hamiltonian Monte Carlo on a transformed space was proposed and then studied empirically. It was found that in some cases little improvement was observed over the

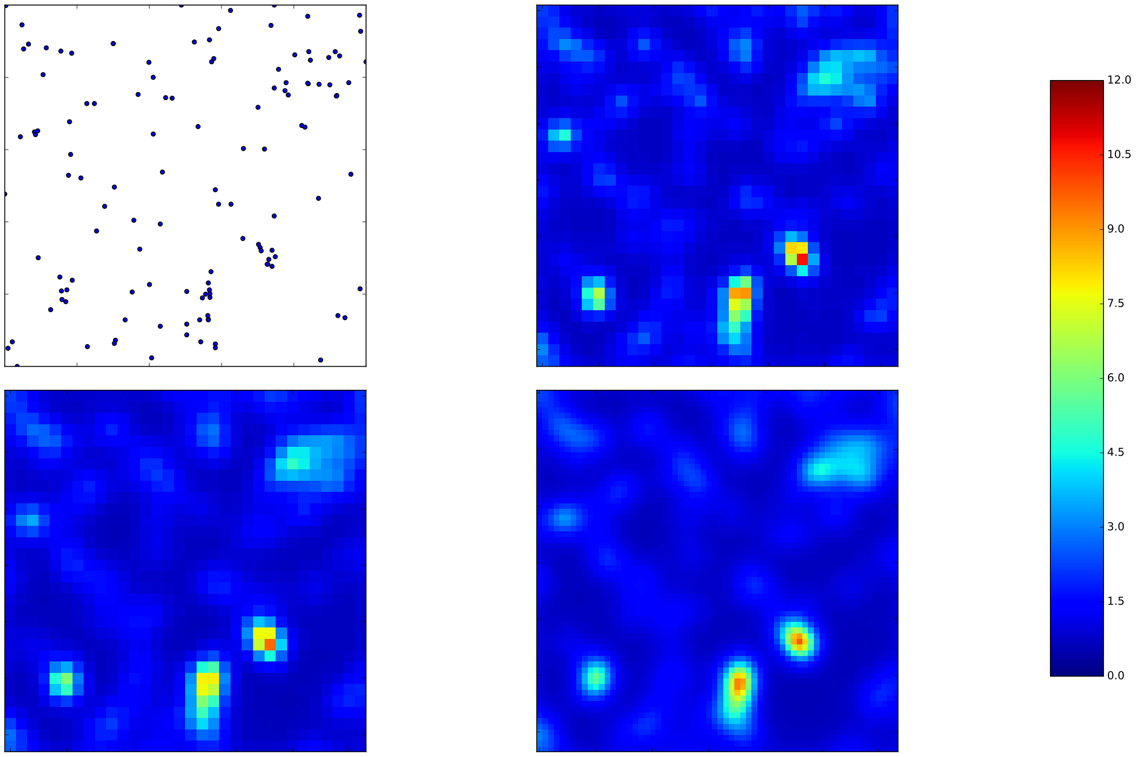

Fig. 5.4 Pine sapling data. From left to right: reported locations of pine saplings; posterior mean intensity on a 32×32 grid using full MCMC; posterior mean intensity on a 32×32 grid with variational MCMC using 225 inducing points, posterior mean intensity on a 64×64 grid using variational MCMC with 225 inducing points.

Gaussian approximation of the previous chapter. In other cases we demonstrated useful qualitative differences from the Gaussian approximations. For example, the application of these ideas in spatial statistics seems promising.

The tight analogy between these variational approximations and inference in hierarchical latent Gaussian models means that any future progress in the latter applies to the former. We look forward to further progress on minibatch MCMC methods as we believe this will be a particularly compelling use case. Similarly, although HMC gave us satisfactory approximations, one could also investigate the use of pseudo marginal MCMC in its place (Filippone and Girolami, 2014).

Taking a step back, we would reflect that contemporary uses of variational inference can now look quite different from some of the traditional approaches discussed in section 2.2. Going forward we hope this will allow us to exploit probabilistic models of still further sophistication.

Chapter 6

GPflow: A Gaussian process library using TensorFlow

Available at https://github.com/gpflow/gpflow

6.1 Introduction

GPflow is a Gaussian process library that uses TensorFlow for its core computations and Python for its front end. The distinguishing features of GPflow are that it uses variational inference as the primary approximation method, provides concise code through the use of automatic differentiation, is able to exploit GPU hardware, and has been engineered with a particular emphasis on software testing. Both GPflow and a version of TensorFlow are available as open source software under the Apache 2.0 license. GPflow is managed by Alexander G. de G. Matthews and James Hensman, whilst TensorFlow is managed by Google.

In this chapter, we will first outline our broad goals for the GPflow project. We will then systematically explain how the features of the software help us to achieve these goals and how our contribution differs from that of other Gaussian process libraries. To implement these key features there are significant gains to be made by using neural network software libraries. Further, we argue that of the available similar packages, TensorFlow is the best choice. We then discuss work we undertook to adapt TensorFlow to the particular needs of Gaussian process algorithms. Having motivated the overall design of GPflow we then explain the software functionality and architecture in more detail. We also outline the quality control practices we use in the project. In timed experiments on a practical example, we show that GPflow gives meaningful speed gains. Finally we conclude.

6.2 Existing Gaussian process libraries

There are now many publicly available Gaussian process libraries ranging in scale from personal projects to major community tools. We will therefore only consider a relevant subset of the existing libraries. A very influential Gaussian process software package is the GPML toolbox originally created by Nickish and Rasmussen (2010). The software works in *MATLAB* or *OCTAVE*. It has concentrated on core functionality, giving a clean implementation of the most fundamental features. It has been widely augmented and forked by third parties, often on an informal basis, for research on Gaussian processes. A key reference for our particular contribution is the GPy library (GPy, 2012), which is written primarily using Python and Numeric Python (NumPy). GPy has an intuitive object oriented interface. Another relevant Gaussian process library is GPstuff (Vanhatalo et al., 2013) which is also a *MATLAB* and *OCTAVE* library. There is considerably more functionality in GPstuff than GPML.

6.3 Objectives for a new library

GPflow is motivated by the following goals:

1. Support for a variety of likelihoods and kernel functions.

2. Accurate approximation of intractable quantities of interest.

3. Speed, particularly on large datasets.

4. Verifiably correct implementation of its component algorithms.

5. An intuitive user interface.

6. Easily extensible code.

We argue that there is a way to better meet these simultaneous objectives than existing packages and that it is realised in GPflow. We will proceed by explaining the reasoning we followed in designing our software to meet these goals.

6.4 Key features for meeting the objectives

To best meet the key goals of our library, we were led to a project that had all of the following distinguishing features:

(i) The use of variational inference as the primary approximation method to meet the twin challenges of non-conjugacy and scale.

(ii) Relatively concise code which uses automatic differentiation to remove the burden of gradient implementations.

(iii) The ability to leverage GPU hardware for fast computation.

(iv) A clean object oriented Python front end.

(v) A dedication to testing and open source software principles.

We now explain how each of these features relate to our stated objectives in turn.

The assertion that the use of variational inference allows us to meet the challenges of non-conjugacy and scale is one of the main claims explained and supported in the rest of this thesis. For instance, see chapter 4 for more discussion of this point. With regards to the specific stated goals of this section, this means that variational inference is a good algorithmic framework to meet objectives 1, 2 and 3.

In the past, much developer time in the variational inference and Gaussian process communities has been spent implementing gradients on a case by case basis. This is because in both cases the algorithms used typically rely heavily on gradient based optimization of an objective function. The development is time consuming and leads to relatively long and complex code. By shortening the code we can make it easier to extend the software (objective 6) and to deliver verifiably correct code (objective 4) through good test coverage. Automatic differentiation is therefore extremely helpful in advancing our objectives.

The utility of commodity graphics processing units (GPUs) for speeding up general scientific applications and other areas of machine learning is well established. In the experiments section 6.7, we will demonstrate that using GPUs can significantly speed up a Gaussian process approximation for a large dataset. Thus using GPUs is important for creating fast software (objective 3).

We argue that a clean object oriented front end is the most natural way to interface with a Gaussian process library. In this paradigm, the progress of a typical Gaussian process modelling task becomes the life cycle of a corresponding Model object. The object can then naturally be given attributes such as data and model parameters. Methods correspond to things we do with or to the Model object during the modelling process such as making predictions or training the parameters. Python is a popular and elegant language for object oriented programming. It is relatively light weight and approachable. By contrast, we argue that *MATLAB* is not well suited to object oriented programming because object oriented

Table 6.1 A summary of the features possessed by existing Gaussian process libraries at the time of writing. OO stands for object oriented. In the GPU column GPLVM denotes Gaussian process latent variable model and SVIGPC is the Stochastic variational inference Gaussian process classification discussed in chapter 4 and in the experiments section 6.7. For more discussion of the comparison between GPy and GPflow, see the text.

Library	**Sparse variational inference**	**Automatic differentiation**	**GPU demonstrated**	**OO Python front end**	**Stated test coverage**
GPML	✗	✗	✗	✗	N\A
GPstuff	Partial	✗	✗	✗	N\A
GPy	✓	✗	GPLVM	✓	49%
GPflow	✓	✓	SVIGPC	✓	99%

concepts are not well implemented. As a result, most *MATLAB* libraries use a procedural paradigm. This proclivity towards the procedural design disadvantages Gaussian process libraries that use *MATLAB* because of the aforementioned advantages of object oriented structure. To summarize, an object oriented Python interface helps in our goals of an intuitive user interface (objective 5) and ease of extensibility (objective 6).

Good test coverage helps both the quality and speed of development on a project. This latter point is counter-intuitive, but we argue that coding on an unsteady foundation is often held up by impenetrable bugs. Research software will be most impactful when the whole process is transparent and verifiable. This means having open source code, publicly verifiable test coverage, and ideally a public record of code reviews. We believe that our dedication to these principles is shared by the developers of other Gaussian process libraries. A big component in our relative success in providing high test coverage is the code reduction we obtained using automatic differentiation (feature (ii)). To reiterate, we view testing and adherence to open source software principles as essential procedures in providing verifiably correct algorithms (objective 4).

We now give a summary of which Gaussian process libraries possess the distinguishing features discussed in table 6.1. Looking at the table we can see that the closest library to our own is GPy. We therefore now discuss the relationship of GPflow to GPy in more detail. The GPflow interface is based on the GPy interface, because, as we have already discussed, we believe an object oriented Python interface to be the best way to interface with a Gaussian process library. Further, we believe that, as an example of such an interface, the GPy implementation is essentially correct. The object oriented GPy architecture influenced that of GPflow, but as we shall see later, an important difference between GPflow and GPy is that GPflow uses TensorFlow for its core computations rather than numeric Python. This difference significantly affects the general requirements of the architecture. The use of GPUs to speed up training for the Gaussian process latent variable model (GPLVM) was discussed in Dai et al (2014). The solution there was targeted towards that specific model and is essentially a CUDA implementation of the kernel expectation computations that constitute the bottleneck of the corresponding training algorithm. It is this code that constitutes the current GPU functionality in GPy. By contrast the GPflow implementation is targeted at a broad variety of GPU capability. Although GPU coverage is not total, much of the key functionality has a GPU implementation. We show considerable speed improvements for a realistic use case in section 6.7, and outline the next steps for further reductions in compute time. At the time of writing, GPy supports some functionality that GPflow does not. For example the GPLVM is not yet implemented in GPflow and GPy has a larger variety of kernels. In this sense the core functionality we have at the moment is closer in spirit to GPML, which as already stated is often extended by the research community. We hope that the emphasis we have placed on extensibility (objective 6) will mean that GPflow is also used in this way.

Having established a desirable set of key design features, the question arises as how best to engineer a Gaussian process library to achieve them. A central insight here is that:

Many of the features we highlight are well supported in neural network libraries.

By ‘neural network libraries’ we mean packages like TensorFlow, Theano and Torch. Backpropagation is one of the standard training tools in the neural network community. Neural network software has made working with neural networks easier by using automatic differentiation (feature (ii)) to reduce the coding overhead for a user. The utility of GPUs for speeding up neural network training is well established and thus the best neural network software packages have sophisticated code for interfacing with such hardware (feature (iii)).

The next section argues that of the available neural network software, TensorFlow is the right choice for our needs.

6.5 TensorFlow

In the last section, we were led to the conclusion that a Gaussian process library could benefit from the use of a neural network library. Our contention is that TensorFlow has a claim on being the best such software in general and is also best suited to the specific needs of Gaussian process software. In this section, we first explain TensorFlow in more depth and then detail our argument for its general and specific utility. Although using TensorFlow led to large benefits in our project, we did have some requirements that were not shared by the neural network community. To this end it was also necessary to contribute new functionality to TensorFlow, as we shall also relate.

The TensorFlow white paper (Abadi et al., 2015), which is the main reference for this section, describes TensorFlow as "an interface for expressing machine learning algorithms, and an implementation for executing such algorithms". This clear delineation is made particularly cleanly in TensorFlow. In TensorFlow a computation is described as a "stateful dataflow graph". The graph is a directed graph where the nodes represent *Operations* (*Ops* for short) and the edges represent *Tensors*. As a directed edge, a Tensor represents the flow of some data between computations. Ops are recognisable mathematical functions such as addition, multiplication etc. *Kernels* (in an unfortunate collision in terminology with the Gaussian process literature) are implementations of a given Op on a specific device such as a CPU or GPU. Further details of the interface can be found in the TensorFlow white paper (Abadi et al., 2015).

The clear separation between interface and implementation means that TensorFlow is very flexible and portable. The Google internal implementation of TensorFlow is not identical to the open source one, although they share an interface. TensorFlow implementations have been run on large multi-server GPU clusters and on mobile devices.

Like almost all modern neural network software, TensorFlow comes with the ability to automatically compute the gradient of an objective function with respect to some parameters. In TensorFlow this is neatly viewed as a function that takes a computational graph corresponding to a forward computation and extends it to perform the desired gradient computation.

The computational graph abstraction is also useful for parallelization. Parallelization can often be visualized as a partitioning of the computational graph into subgraphs and then 'fixing' the breaks with special messaging Ops, that may for instance represent communication between servers. TensorFlow has heuristics to perform this partitioning automatically and allocate the sub-graphs to the available hardware.

The TensorFlow open source implementation comes with a wealth of GPU kernels for the majority of Ops. This code represents many months of developer activity and is an important resource for the larger community. The combination of device level parallelization and GPU functionality allows TensorFlow graphs to be scalable to hundreds of GPUs.

Since Google released an open source version of TensorFlow there has been widespread uptake. The TensorFlow open source release has been the most asked about package in its category on Stack Overflow since its release (Rao, 2016). TensorFlow was the most forked of any repository on GitHub in 2015, despite being released in November of that year (Rao, 2016). Google has continued to manage and contribute to the repository. This has the effect of ‘anchoring’ the codebase and enforces exacting coding standards.

6.5.1 The comparative advantages of using TensorFlow

Having argued in section 6.4 that it was beneficial to use neural networks libraries within Gaussian process software, we here compare TensorFlow to other similar software. We will consider the main alternative packages in turn.

Theano (Theano Development Team, 2016) is, in many ways, similar to TensorFlow. Historically Theano came first and in fact some of the original developers of Theano moved to Google to develop TensorFlow. It is possible that projects will continue to use Theano for legacy reasons. For us the crucial factors in choosing TensorFlow over Theano were the shorter compile times and the excellent TensorFlow distributed code. Torch (Collobert et al., 2002) is a strong competitor to TensorFlow and has strong industrial support from the likes of Facebook and Twitter. For us the crucial factor in choosing TensorFlow over Torch was that we felt Python would be a better language for the user interface than Lua. Although not generally classified as a neural network package, the Stan Math library (Carpenter et al., 2015) has much in common with TensorFlow since it implements reverse mode differentiation and is written in C++ and Eigen. The greater distributed and GPU computing functionality was what lead us to choose TensorFlow over this library. HIPS Autograd is a library for: "differentiating native Python and Numpy Code" (Maclaurin et al., 2016). Whilst working in native Python is appealing, distributed and GPU computing functionality was again the key differentiator.

6.5.2 Contributing GP linear algebra requirements to TensorFlow

Although we gained significantly from using TensorFlow within GPflow, there were some linear algebra capabilities that were not yet present in the software which were required for our purposes. We therefore added this functionality to TensorFlow. In this section, we explain the linear algebra requirements and give an overview of the solution.

Gaussian process software needs the ability to solve systems of linear equations using common linear algebra algorithms. Take, as a simple example, the exact log marginal likelihood $\log p(\mathbf{y}|\theta, \epsilon^2)$ of a Gaussian process model with kernel K, zero mean function and a Gaussian likelihood

$$\log p(\mathbf{y}|\theta, \epsilon^2) = \log \mathcal{N}(\mathbf{y}|0, K_{D,D}(\theta) + \epsilon^2 I) \tag{6.1}$$

$$= -\frac{1}{2}\mathbf{y}^T(K_{D,D}(\theta) + \epsilon^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K_{D,D}(\theta) + \epsilon^2 I) - \frac{N}{2}\log(2\pi)\,. \tag{6.2}$$

Here $\mathbf{y}$ is a vector of observed outputs, θ represents the kernel hyperparameters, ϵ^2 is the variance of the Gaussian likelihood and N is the number of data points. Using the notation of section 1.2.3, $K_{D,D}$ is a square kernel Gram matrix for the data inputs D and as such is a function of the covariance hyperparameters θ. For what follows we will not always show functional dependence on the noise variance ϵ^2 and the covariance hyperparameters θ explicitly. This will allow us to keep the equations compact.

A basic requirement of a Gaussian process library is the ability to maximize the log marginal likelihood $\log p(\mathbf{y}|\theta, \epsilon^2)$ with respect to θ and ϵ^2. This is known as type-II likelihood maximization and is discussed in section 1.2.4. To do this we need to be able to compute the log marginal likelihood and its derivatives with respect to the likelihood and covariance hyperparameters.

To compute the log marginal likelihood in a numerically stable way, a common method uses the Cholesky decomposition of the symmetric positive definite matrix $\mathbf{K} = K_{D,D} + \epsilon^2 I$, which satisfies the relation

$$LL^T = \mathbf{K}\,. \tag{6.3}$$

We then define the vector α by

$$\alpha = L^{-1}\mathbf{y} \tag{6.4}$$

and the scalar β by

$$\beta = \log\det\mathbf{K}\,. \tag{6.5}$$

The log marginal likelihood is then related to these quantities by the equation

$$\log p(\mathbf{y}|\theta, \epsilon^2) = -\frac{1}{2}\alpha^T\alpha - \frac{1}{2}\beta - \frac{N}{2}\log(2\pi)\,. \tag{6.6}$$

The vector α may be stably computed by solving the linear equation

$$L\alpha = \mathbf{y} \tag{6.7}$$

```
import tensorflow as tf

#Inputs:
# y representing observed data.
# Kdd representing the data Gram matrix.
# N representing the number of data points.
# variance representing the Gaussian noise variance.

#Outputs:
# logP representing the marginal likelihood.

K = Kdd + variance * eye(N)
L = tf.cholesky(K)
alpha = tf.matrix_triangular_solve(L, y, lower=True)
beta = tf.reduce_sum(tf.log(tf.diag_part(L)))
logP = -0.5*tf.reduce_sum(tf.square(alpha))-0.5*beta-0.5*N*tf.log(2.*pi)
```

Fig. 6.1 A simple TensorFlow implementation of the log marginal likelihood of a Gaussian process with a Gaussian likelihood, as discussed in section 6.5.2. In conjunction with the text of that section, most operations used are self explanatory, except perhaps reduce_sum which sums over all input elements. In GPflow, the equivalent code is distributed across multiple functions for architectural reasons, as will be discussed in section 6.6.2.

for α using forward substitution. The scalar β can also be computed using the Cholesky matrix by using its definition (6.5) and noting that

$$\log\det\mathbf{K} = 2\sum_{i=1}^{N}\log L_{i,i}\,. \tag{6.8}$$

Simple Python TensorFlow code for specifying the log marginal likelihood computational graph in this way is shown in figure 6.2. As we will discuss later in section 6.6.2, the equivalent GPflow source code for this example is distributed across multiple functions for architectural reasons. This example demonstrates that the ability to perform linear algebra operations such as the Cholesky decomposition and to solve triangular systems of linear equations using forward substitution is an important component of Gaussian process libraries.

If each Op in the TensorFlow code (figure 6.2) for the log marginal likelihood has gradient code, then we can compute the parameter derivatives of the objective automatically. TensorFlow was open sourced with many common linear algebra Ops, but without the ability to differentiate computational graphs using them in all cases. Much of the necessary functionality can be implemented efficiently using the existing Ops and was contributed by the community over the following months. However, the differentiation of computational graphs that used the Cholesky decomposition required a new Op, which we contributed with the help of Rasmus Munk Larsen at Google, who reviewed the code.

We briefly discuss the Cholesky gradient algorithm at a high level. Whilst there are analytic linear algebra expressions for the backpropagated derivatives of a Cholesky decomposition, for matrices of a reasonable size they are slower than an alternative family of approaches (Murray, 2016b). Starting with the work of Smith (1995) the idea is to take an implementation of the Cholesky decomposition and differentiate it line by line. It turns out this can be done efficiently, in-place in memory, and using vectorized operations. Initially, Smith differentiated the unblocked Cholesky algorithm. The blocked Cholesky decomposition, that is used in most modern linear algebra implementations can be much faster on a modern CPU. Murray (2016b), who also gives an extensive literature review, suggested differentiating this more complex algorithm, which leads to a faster backpropagation algorithm. It is this algorithm that we implemented in C++ and Eigen, and then integrated into TensorFlow. The code, along with some comments is included in Appendix F.

The inclusion of Cholesky backpropagation in TensorFlow means that the Cholesky decomposition Op can be composed with other Ops which have gradients, to make a computational graph that is differentiable overall. As we have already noted, the Cholesky decomposition is a necessary step in the preferred method for solving linear systems in Gaussian process libraries, so this removes one of the main barriers to Gaussian process inference in TensorFlow.

6.6 GPflow

Having discussed the motivation and high level design concepts of GPflow, along with the fundamental role that TensorFlow plays in achieving our goals, we now give more detail about the functionality and architecture of GPflow. We then detail procedures to ensure quality and usability of our software.

6.6.1 Functionality

GPflow supports exact inference where possible, as well as a variety of approximation methods. One source of intractability is non-Gaussian likelihoods, so it is helpful to categorize the available likelihood functionality on this basis. Another major source of intractability is the adverse scaling of GP methods with the number of data points. To this end we support 'variationally sparse' methods which use prior conditional matching (see section 2.2.4) to guarantee that the approximation is scalable and close in a Kullback-Leibler sense to the posterior. Whether or not a given inference method uses variational sparsity is another useful way to categorize it.

Table 6.2 A table showing the inference classes in GPflow and the model options they support. The details are discussed in the text.

	Gaussian likelihood	**Non-Gaussian likelihood (variational)**	**Non-Gaussian likelihood (MCMC)**
Full covariance	GPR	VGP	GPMC
Variational sparsity	SGPR	SVGP	SGPMC

The inference options, which are implemented as classes in GPflow, are summarized in table 6.2. We now give more detail on each class in the order they appear in the table. The GPR class implements exact Gaussian process regression with a Gaussian likelihood. The VGP class implements variational inference without sparsity for non-Gaussian likelihoods, based on the method of Opper and Archambeau (2009). The GPMC class allows MCMC without any sparsity assumptions. The SGPR class implements the Titsias method for sparse variational regression (2009a), which is also discussed in detail in chapter 3. The SVGP class is the sparse variational approximation discussed in chapter 4. SGPMC implements the variationally sparse MCMC discussed in chapter 5. Note that all the MCMC based inference methods support a Bayesian prior on the hyperparameters, whereas all other methods assume a point estimate. Our main MCMC method is Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2010). Conveniently, we can obtain all the necessary gradients for this algorithm automatically using TensorFlow.

A variety of likelihood functions are implemented in GPflow . At the time of writing we have Gaussian, Poisson, Exponential, Student-T, Bernoulli, Gamma, Beta and Multinomial likelihoods. One of our goals at the outset of the project was extensibility (section 6.3). As discussed in chapter 4, sparse variational inference with a variety of smooth likelihoods is possible. The minimal requirement in GPflow for supporting a new likelihood is to supply TensorFlow functions that return $p(y|f_d)$ and its logarithm. The base requirements for likelihoods using the MCMC based methods are the same. Likelihoods in GPflow are implemented by inheritance from a base class, which take these functions and perform Gauss-Hermite quadrature where necessary for variational inference. Optionally, these Gaussian quadratures may be overridden in daughter classes with an analytic expression, if it is available. There is no need to implement gradients for likelihood functions.

Kernels supported in GPflow include the stationary RBF, exponential, and Matérn kernels, as well as the non-stationary cosine, periodic, and linear kernels. We also support kernels that are compositions of base kernels through addition or products. Implementing a new kernel is again very straight forward. The inference methods in GPflow do not make strong assumptions about the kernel Gram matrices. Since we can use automatic differentiation there is no need to write gradient code. We demonstrate this simplicity by showing GPflow Python source code for the linear kernel in figure 6.2. The kernel implements the following functionality

$$K(x^{(1)}, x^{(2)}) = \sigma_i^2 \sum_i x_i^{(1)} x_i^{(2)} . \tag{6.9}$$

If automatic relevance determination (ARD) is activated then there is a variance parameter σ_i^2 for each input dimension, otherwise there is a single shared variance parameter. Common kernel functionality is inherited from a base class in line 1. The key code that needs to be written for a new kernel is a TensorFlow implementation of the Gram matrix computation (line 23) and a way to compute the diagonals of the Gram matrix (line 30). The class Param used in lines 18 and 20 will be explained in section 6.6.2.

6.6.2 Architectural considerations

As already stated, the GPflow interface is very similar to that of the open source GPy. GPy also had some influence on the general architecture of GPflow. For more discussion of the similarities and differences between the two packages, see section 6.4. The whole Python component of GPflow is intrinsically objected-oriented. The code for the various inference methods in table 6.2 is structured in a Class hierarchy, where common code is pulled out into a shared base class. Typically, the classes at the bottom of the hierarchy have TensorFlow code that expresses key quantities in a TensorFlow graph.

As an example, consider the code extract from the Gaussian process regression (GPR) class in figure 6.3. For the present discussion, it is not essential to understand all aspects of this code -we will consider only those key features of the code which are currently relevant. As can be seen in line 1 of the figure, the GPR class inherits from a GPmodel class which carries some more generic code. The constructor definition in line 9 takes input data X and output data Y as well as kernel and mean function objects. These are stored for use during the life cycle of the GPR object. The function *build_likelihood* starting on line 21 defines a TensorFlow computational graph for the log marginal likelihood $\log p(\mathbf{y}|\theta, \epsilon^2)$ defined in equation (6.1). The function calls a subroutine *multivariate_normal* which returns a computational graph for the relevant Gaussian density.

The object-oriented paradigm, whilst very natural for the Python layer of the code, has a different emphasis to that of a computational graph. As a directed graph, the computational graph is arguably closer to a functional concept. There is some ability to hold persistent

```python
class Linear(Kern):
    """
    The linear kernel
    """
    def __init__(self, input_dim, variance=1.0, active_dims=None, ARD=False):
        """
        - input_dim is the dimension of the input to the kernel
        - variance is the (initial) value for the variance parameter(s)
          if ARD=True, there is one variance per input
        - active_dims is a list of length input_dim which controls
          which columns of X are used.
        """
        Kern.__init__(self, input_dim, active_dims)
        self.ARD = ARD
        if ARD:
            # accept float or array:
            variance = np.ones(self.input_dim)*variance
            self.variance = Param(variance, transforms.positive)
        else:
            self.variance = Param(variance, transforms.positive)
        self.parameters = [self.variance]

    def K(self, X, X2=None):
        X, X2 = self._slice(X, X2)
        if X2 is None:
            return tf.matmul(X * self.variance, tf.transpose(X))
        else:
            return tf.matmul(X * self.variance, tf.transpose(X2))

    def Kdiag(self, X):
        return tf.reduce_sum(tf.square(X) * self.variance, 1)
```

Fig. 6.2 A GPflow implementation of a linear kernel using the Python programming language. The details are discussed in section 6.6.1.

```
class GPR(GPModel):
    """
    Gaussian Process Regression.

    This is a vanilla implementation of GP regression with a Gaussian
    likelihood.  Multiple columns of Y are treated independently.

    """
    def __init__(self, X, Y, kern, mean_function=Zero()):
        """
        X is a data matrix, size N x D
        Y is a data matrix, size N x R
        kern, mean_function are appropriate GPflow objects
        """
        likelihood = likelihoods.Gaussian()
        X = DataHolder(X, on_shape_change=’pass’)
        Y = DataHolder(Y, on_shape_change=’pass’)
        GPModel.__init__(self, X, Y, kern, likelihood, mean_function)
        self.num_latent = Y.shape[1]

    def build_likelihood(self):
        """
        Construct a tensorflow function to compute the likelihood.

        \log p(Y | theta).

        """
        K = self.kern.K(self.X)+eye(tf.shape(self.X)[0])*self.likelihood.variance
        L = tf.cholesky(K)
        m = self.mean_function(self.X)

        return multivariate_normal(self.Y, m, L)
```

Fig. 6.3 An extract from the Python GPR class which implements Gaussian process regression in GPflow. The details are discussed in section 6.6.2.

state in a TensorFlow computational graph through Variables and this takes it away from a pure functional construct. These two programming paradigms, namely a largely functional computational graph, and a object-oriented interface need to live cleanly together in GPflow. This is achieved through the Param class that allows parameters to be both properties of an object that can be manipulated as such and Variables in a TensorFlow graph that can be optimized.

6.6.3 Project quality and usability considerations

To help with our goal of verifiable correct software we have a number of project rules and procedures.

All GPflow source code is openly available on GitHub. This means that science done with GPflow is reproducible and transparent. This open attitude extends to the version history of the code, which is also all publicly available.

The web page uses continuous integration to run an automated test suite. Both the result of the automated tests and the test coverage are publicly visible on the front page. As already mentioned in table 6.1 and section 6.4, the test code coverage for GPflow is higher than similar packages where the code coverage statistics are published, achieving a level of 99%. The addition of new code to the GPflow master branch can only be done by a pull request. Our infrastructure ensures that it is not possible to merge a pull request that reduces the test coverage or fails the tests. One reason that we have been able to achieve high test coverage is the reduction in code that comes with not having to implement gradients by hand. We regard test coverage as a necessary, but not sufficient condition for good testing of a code base. Care is taken to make sure that tests check functionality rather than just run the code. The majority of tests are unit tests that check correct implementation of a certain mathematical function. There are also scenario style tests that check a lot of functionality at once. For instance, when there are as many inducing points as data points and the inducing inputs equal the data inputs, sparse variational Gaussian process regression and exact Gaussian process regression should give identical results. Such sanity checks are useful because they test a lot of complex functionality at once.

The project has a culture of code review. By agreement, a pull request may not be merged by a major contributor of code to that change. Instead, another GPflow developer reviews the code and makes a decision as to whether to merge. The discussion of the pull request may be seen on a web page and anyone may comment on potential new code.

A user manual for GPflow can be found at *http://gpflow.readthedocs.io* . The documentation is generated from Jupyter notebooks made during the development of GPflow by the contributors. It includes both example code and plots of the results. As such it is a good next port of call for people who want to use GPflow.

6.7 Experiments

In this section, we show that for a real use case our design decisions have enabled us to deliver on our goal of relatively fast software. Code for these timing experiments can be found at *https://github.com/alexggmatthews/GPflow_profiling*.

As a scenario, we chose to study the training algorithm from this thesis that took the longest, namely optimizing the multiclass MNIST classifier of section 4.5.5 using stochastic variational inference. We compared against the popular GPy software, which, as discussed in section 6.4, is the closest in terms of features to GPflow. None of the other libraries discussed support this algorithm.

Functionally, the algorithms are nearly identical in GPflow and GPy. In both cases we used the AdaDelta optimizer. The optimizer implementations required us to recalibrate the step sizes. Running the two implementations for the same number of cycles we obtained accuracies within 0.1% of one another, despite the stochasticity of the algorithm. The minibatch size and number of inducing points were the same in both implementations and were unchanged from section 4.5.5.

Under controlled conditions, we did a series of trials, on each occasion measuring the time each package took to perform 50 iterations of the algorithm. For each set of trial conditions we repeated the experiment five times. To best counter any systematic bias from possible drift in base compute speed, we interleaved the trials of the two packages and the various experimental setups. The trials included a set of CPU experiments, where we varied the number of threads available to the two packages. For GPflow, we also measured the effect of adding a GPU on top of the maximal number of CPU threads considered. GPy does not presently have a GPU implementation of this algorithm.

GPy was set up as follows. We used the recent stable release *v1.4.1*. For the multiclass likelihood we used our own Cython implementation, which we heavily optimized for a previous paper. We confirmed that we were using the Intel Math Kernel Library optimizations of numeric Python, that are released with the popular Anaconda Python bundle.

We now detail our GPflow settings. We use GPflow branch *profiling_mods*, which was taken from the master branch and modified to allow us to manipulate the number of TensorFlow threads. We used float32 as the main TensorFlow real valued data type.

The hardware used was a single Linux research workstation, which had both CPUs and a GPU. We used an Intel Core I7-4930K CPU clocked at 3.40GHz. This model has six true cores with an additional six hyper threads. In our experience, a hyperthread cannot be treated as a totally independent entity with intensive numerical code, so we conservatively limited ourselves to using a maximum of six threads in the experiments. The GPU was an NVIDIA GM200 Geforce GTX Titan X. None of the experiments reached any of the system memory limits.

The results of the timing experiments are shown in figure 6.4. We will consider the CPU comparison first. For one to four CPU threads, the speeds for GPflow and GPy are very similar. Given that efforts have been made to optimize both packages, this possibly means that in this case both libraries are near the full potential of the hardware. For five and six threads, we note that GPy has a lower mean speed with a high variance. We suggest that this could be a result of occasional difficulties synchronizing the threads. For both packages we note that there are diminishing returns in using additional CPUs. The GPU results are also shown on figure 6.4. It can be seen that the increase in speed from adding a GPU is considerable.

Having given a detailed technical analysis of the speed of GPflow, it is also useful to consider the practical implications of this work for research. The MNIST training algorithm uses 200000 iterations of stochastic variational inference. Based on the measurements we have made, training using GPflow with 6 CPU threads would take approximately 41 hours or just under 2 days. Adding a GPU would currently reduce the training time to about 5 hours. These gains could make a significant difference to the work flow of a researcher on this topic and allow avenues of research that would not otherwise have been practical.

6.8 Conclusion

In this chapter we have introduced GPflow, a new Gaussian process library which uses TensorFlow.

We discussed our goals for the project, which we believe we have achieved. We offer a broad variety of core GP functionality, for instance supporting a variety of likelihoods and kernels. We use recent research progress to approximate intractable quantities of interest. In this direction, we have demonstrated empirically that the software is relatively fast-particularly on large datasets. We provide an intuitive user interface and our code is easily extensible by other researchers. Based on visible quality assurance procedures, people using or extending the code gain confidence that it functions correctly.

We achieved our objectives by identifying and pursuing the key distinguishing features which we motivated in this document. Variational inference provides us with an unified and flexible approximation framework. The use of automatic differentiation enabled us to reduce

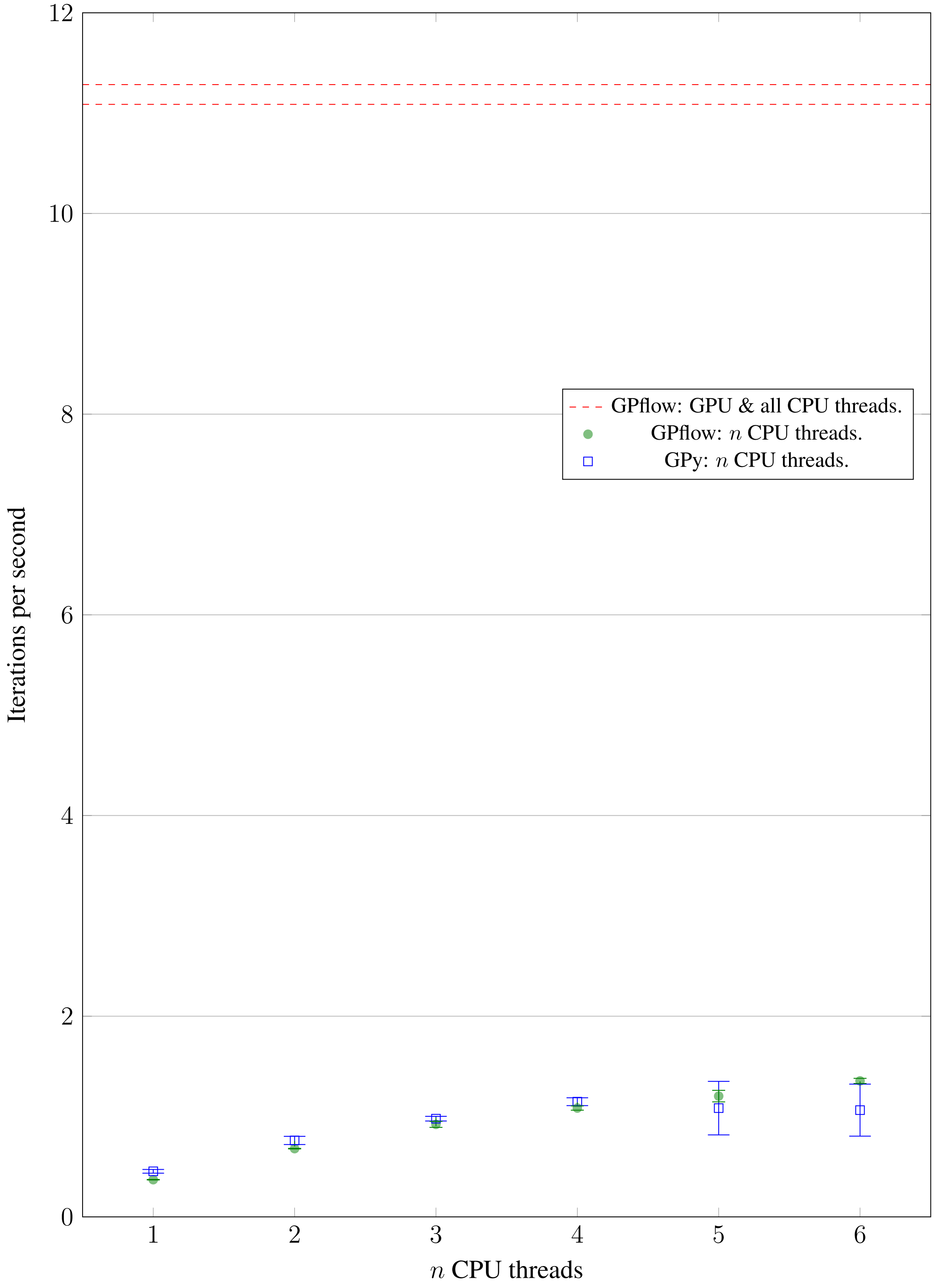

Fig. 6.4 A comparison of iterations of stochastic variational inference per second on the MNIST dataset for GPflow and GPy. Error bars shown represent one standard deviation computed from five repeats of the experiment.

the amount of code necessary for the project. We exploited GPU hardware to significantly speed up our algorithms. Using the Python programming language enabled an intuitive object oriented interface to the software. Throughout the development of the code we followed the principles of open source software development, for instance achieving 99% test coverage of our code. Having identified these features, a review of the available Gaussian process libraries showed that no existing package has them all.

Engineering these features was made easier by the use of neural network software. Of the available packages of this type, we chose TensorFlow on the basis of a systematic comparison. Although TensorFlow helped our project considerably, we needed to add some linear algebra functionality used in Gaussian process libraries. In particular, we contributed reverse mode differentiation for the Cholesky decomposition to the project.

Having described the successful design process for GPflow, we then elaborated on the specifics of the software. The functionality was elucidated in more detail. We explained the architecture we chose and how quality and usability is ensured on the project.

Finally in systematic timed experiments, we compared GPflow to GPy on a real use case, namely that of stochastic variational inference training on the large MNIST dataset. We found that for small numbers of CPU threads GPflow and GPy have similar performance, although GPy showed some evidence of thread synchronization issues as the number of threads was increased. GPflow is currently the only package that can exploit GPU hardware for this algorithm and we showed that the consequent speed gains are large enough to qualitatively improve the workflow for research in this area.

In terms of further work on the project, it will be divided into optimizing performance and introducing new functionality. The immediate priority in the performance direction is to work with the TensorFlow community to provide GPU kernels for more of the Ops we use. As this work progresses, it will open up the possibility of using multiple GPUs in parallel to further reduce execution time.

New functionality will be divided into new features for the core library, new libraries from the GPflow team that depend on the core library and, we hope, more projects that use GPflow in the broader community. A big likely growth area for new functionality is to support more complex models which are defined using Gaussian processes as a component. Examples include the Gaussian process latent variable model (Lawrence, 2003), the deep Gaussian process (Damianou and Lawrence, 2013) and the Gaussian process state space model (Frigola et al., 2014).

Chapter 7

Conclusion

7.1 Contributions

In this thesis we have studied the general problem of scalable variational inference in Gaussian process models. We have also contributed a new Gaussian process software library that includes these methods. Here we briefly summarize the main contributions.

Chapter 3 reconsidered the foundations of sparse variational approximations. The popular sparse variational approximation of Titsias (2009a) uses an optimization objective that is motivated using an augmentation argument. Such arguments state that variational inference in an augmented model, which includes augmentation parameters, is equivalent to variational inference in the original model. Under a broad definition of equivalence, we showed that such arguments are false. Fortunately, as we showed, the same optimization objective used in Titsias (2009a) can be rigorously derived under general conditions by considering a $\mathcal{KL}$-divergence between the approximating process and the posterior process. This argument, which built on the prior work of Seeger (2003a; 2003b), therefore justifies the Titsias approximation framework which works well in practice. To our knowledge this connection was unremarked in the literature prior to our work. We then used this theoretical framework to characterize broad conditions under which adding additional inducing points will yield a monotonic improvement in terms of $\mathcal{KL}$-divergence. This is a property that many other sparse approximations do not have. We described the use of *variational inducing random variables* which allow us to work with more expressive approximating families, whilst maintaining the posterior process as the target distribution. We showed that variational interdomain approximations follow very naturally from this concept, whereas it is difficult to justify such extensions using an augmentation argument. Another example given of the explanatory power of the new theory is its ability to resolve challenging issues around applying sparse variational inference in models like Cox processes where the likelihood depends on infinitely many function points.

Chapter 4 addressed two commonly encountered challenges of inference in Gaussian processes simultaneously:

1. Approximate integration in non-conjugate Gaussian process modelling using classification as the main focus.

2. The adverse $\mathcal{O}(N^3)$ scaling with the number of input data points.

The fundamental nature of these two questions has lead to a rich prior literature. A critical review of this literature showed that a number of key features emerge that need to be understood and integrated to obtain a competitive method. The features are as follows. The method must be able to deal with the non-analytic posterior associated with a non-Gaussian likelihood. Use of an M inducing point sparse method where $M \ll N$ reduces computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$. Where the underlying index set is continuous it is very beneficial to allow inducing inputs that are not data inputs, which takes us beyond the so called 'subset of data' methods. In cases where an order $\mathcal{O}(NM^2)$ method is still prohibitively slow because N is large, it is often possible to reduce the computational load by using a minibatch estimate of the training gradient. The challenge is to do this in a convergent way.

We showed that one unifying framework allows us to achieve all the desired features, namely sparse variational inference. Whereas many previous attempts in this direction have relied on questionable augmentation and opaque bounding arguments the $\mathcal{KL}$-between processes view of chapter 3 allows us to give a lucid derivation of a method that works well in practice. In extensive experiments, we showed that the resulting algorithm gives competitive performance with the popular Generalized FITC approach. We demonstrated that the method can be scaled to give accurate results on datasets like the MNIST digit benchmark, which had been too challenging for Bayesian nonparametric methods.

Chapter 5 gave an example of how the theoretical view advocated in chapter 3 can lead to principled generalizations of sparse variational inference. It highlighted the concept of *variational potentials*, which allows interpretable analogies to familiar graphical model concepts. This insight clarifies how to relax the strong Gaussianity assumption on the inducing output distribution that was made in chapter 4. Since the implied optimal inducing output distribution is not analytically tractable we used MCMC to draw samples instead. The result was an MCMC/variational hybrid method that has a flexible approximating distribution whilst maintaining the $\mathcal{O}(NM^2)$ scaling characteristic of sparse methods. It also enabled us to extend the models we can approximate to include Gaussian process models which have a

Bayesian prior on the hyperparameters. We investigated Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2010) as a specific MCMC algorithm and gave a recipe for using the method in practice.

In a series of empirical examples, we showed cases where variational MCMC gave qualitatively better approximations than the method of chapter 4. In other cases, we found that the simpler Gaussian approximation worked similarly well, which in those situations serves to validate the stronger assumptions the earlier method makes. We also gave experimental comparisons of the variational MCMC method to full $\mathcal{O}(N^3)$ MCMC, finding that the hybrid method gave good agreement, with significantly faster computation.

Chapter 6 was about GPflow, a new Gaussian process library. Influenced by the contributions of the previous chapters in this work, we adopted variational inference as a key approximation method. The project was significantly helped by using TensorFlow an open source computational graph project to which we ourselves contributed linear algebra code. A Python code layer for the software has an intuitive GPy-like (GPy, 2012) interface and an elegant object oriented architecture, as we detailed. Using automatic differentiation led to a substantial reduction in the code required. Adherence to open source principles has allowed us to ensure the quality of our software contribution. For instance the code has 99% unit test coverage. The utility of GPU hardware to drastically speed up variational inference on a large dataset was demonstrated in timed experiments. The progress detailed led us to conclude that GPflow has the potential to improve the work flow for research on Gaussian processes, particularly at scale.

7.2 Some general considerations

7.2.1 A new theoretical beginning

It is humbling to think that much of chapter 3 could have been understood by Andrey Kolmogorov, who's early contributions to probability theory included fundamental work on stochastic processes and information theory[1]. From a broad perspective, the importance of theorems like 2.ii in chapter 3 is that they provide a confluence between a line of work mostly done in the field of machine learning and the larger mathematics literature. The new connection provides many research opportunities. Interdomain inducing points seem to be an under used tool and it would be interesting to consider other continuous linear functionals. The extensive literature considering Gaussian processes from the perspective of Hilbert spaces may well be useful here.

[1] See for instance his collected works (Kolmogorov et al., 1992).

Since the inception of Bayesian nonparametrics, we have known that it was possible to work with infinite dimensional models with finite computation. The challenge of modelling increasingly large datasets necessarily leads us to consider a related issue: what are the general implications for Bayesian nonparametrics if the computation has to not only be finite but scalable? In this work, we have demonstrated examples where a good approximation can be found to the exact Bayesian posterior process at scale. Since the algorithms we used are very well specified theoretically, this raises the question of what quality guarantees we might derive for them. Seeger (2003b) has studied frequentist guarantees on Bayesian nonparametric models, using the PAC Bayes theorems, but we speculate that this general area may now be ready to be taken further.

7.2.2 A third challenge

Further to the two challenges discussed in section 7.1, there is a third challenge for the Gaussian process community that I wish to highlight before closing:

3. To achieve good generalization with large high dimensional datasets.

This is, of course, not generally possible, but has been demonstrated in important real world cases such as image classification benchmarks. It is currently an area of relative strength for deep neural networks. For example, whilst the results we report for the MNIST dataset in section 4.5.5 extended the scale at which accurate Bayesian nonparametric inference can be done, the accuracy achieved is still well behind neural network results without convolutional assumptions.

This is a problem with a long history. Neal's work (1996) showing that certain Bayesian neural networks correspond to Gaussian processes in the limit of large width led people like Rasmussen and Williams (2006) to study them, motivated by a route to inference in flexible models that was relatively tractable and based on sound principles of probability. This seems to us to still be a good reason for researching these models. The question of whether some important representational power is lost in taking this limit was memorably summed up by MacKay (2002), who asked "did we throw out the baby with the bath water?". One observation we would like to make is that the difficult two challenges of section 7.1 have hindered empirical research on these questions. One cannot hope to investigate the relationship between Gaussian processes, their deep extensions and representation learning without an approximation framework for GPs that is accurate, scalable and well understood. We hope that the ideas in this thesis may help in this direction as well.

References

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

Adler, R. J. (1981). *The geometry of random fields*. Society for Industrial and Applied Mathematics.

Álvarez, M. A. (2011). *Convolved Gaussian process priors for multivariate regression with applications to dynamical systems*. PhD thesis, University of Manchester.

Álvarez, M. A., Luengo, D., Titsias, M. K., and Lawrence, N. D. (2010). Efficient multioutput Gaussian processes through variational inducing kernels. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 25–32.

Bauer, M. S., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding Probabilistic Sparse Gaussian Process Approximations. *arXiv preprint 1606.04820.*

Bayer, J., Osendorfer, C., Diot-Girard, S., Ruckstiess, T., and Urban, S. (2016). Climin - a Pythonic framework for gradient-based function optimization. Technical report, Technical University of Munich.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby computational neuroscience unit.

Billingsley, P. (1995). *Probability and Measure*. John Wiley & Sons, Third edition.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer publishing.

Brown, R. (1828). A brief account of microscopical observations made in the months of June, July and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *Philosopical Magazine.*, pages 161–173.

Cao, Y., Brubaker, M. A., Fleet, D. J., and Hertzmann, A. (2013). Efficient optimization for sparse Gaussian process regression. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 1097–1105. Curran Associates, Inc.

Capinski, M. and Kopp, P. (2004). *Measure, Integral and Probability*. Springer Undergraduate Mathematics Series. Springer London.

Carlin, B. and Louis, T. (2010). *Bayes and Empirical Bayes Methods for Data Analysis, Second Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015). The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. *arXiv preprint 1509.07164*.

Chai, K. M. A. (2012). Variational Multinomial Logit Gaussian Process. *J. Mach. Learn. Res.*, 13(1):1745–1808.

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *ICDAR*, pages 1250–1254.

Collobert, R., Bengio, S., and Marithoz, J. (2002). Torch: A modular machine learning software library.

Cox, R. T. (1946). Probability, Frequency and Reasonable Expectation. *American Journal of Physics*, 14(1):1–13.

Csató, L. (2002). *Gaussian processes: iterative sparse approximations*. PhD thesis, Aston University.

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

Dai, Z., Damianou, A., Hensman, J., and Lawrence, N. (2014). Gaussian process models with parallelization and gpu acceleration. *arXiv preprint 1410.4984*.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian Processes. In Carvalho, C. and Ravikumar, P., editors, *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 207–215.

Damianou, A. C., Titsias, M. K., and Lawrence, N. D. (2015). Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research (JMLR)*, 2.

De Finetti, B. (1974). *Theory of probability: a critical introductory treatment*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley.

Doob, J. (1953). *Stochastic Processes*. Wiley Publications in Statistics. John Wiley & Sons.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195:216 – 222.

Efron, B. (2010). *Large-scale inference : empirical Bayes methods for estimation, testing, and prediction*. Institute of mathematical statistics monographs. Cambridge University Press, Cambridge.

Einstein, A. (1905). Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik*, 322:549–560.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Figueiras-Vidal, A. and Lázaro-Gredilla, M. (2009). Inter-domain Gaussian processes for sparse inference using inducing features. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1087–1095. Curran Associates, Inc.

Filippone, M. and Girolami, M. (2014). Pseudo-marginal Bayesian inference for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2214–2226.

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

Frigola, R., Chen, Y., and Rasmussen, C. (2014). Variational Gaussian Process State-Space models. In *Advances in Neural Information Processing Systems 27*, pages 3680–3688. Curran Associates, Inc.

Gal, Y., van der Wilk, M., and Rasmussen, C. E. (2014). Distributed variational inference in sparse Gaussian process regression and latent variable models. In *NIPS*.

Ghahramani, Z. (2012). Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1984).

Ghahramani, Z. and Beal, M. J. (2000). Graphical models and variational methods. In Opper, M. and Saad, D., editors, *Advanced Mean Field Methods: Theory and Practice*. MIT Press.

Ghosh, J. and Ramamoorthi, R. (2003). *Bayesian Nonparametrics*. Springer Series in Statistics. Springer.

Girolami, M. and Rogers, S. (2006). Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18(8):1790–1817.

Google (2016). Google C++ style guide. https://google.github.io/styleguide/cppguide.html.

GPy (since 2012). GPy: A Gaussian process framework in Python. http://github.com/SheffieldML/GPy.

Gray, R. M. (2011). *Entropy and Information Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 2 edition.

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for Big Data. In *Conference on Uncertainty in Artificial Intellegence*, pages 282–290.

Hensman, J. and Lawrence, N. D. (2014). Nested Variational Compression in Deep Gaussian Processes. *arXiv preprint 1412.1370.*

Hensman, J., Matthews, A. G. d. G., Filippone, M., and Ghahramani, Z. (2015a). MCMC for variationally Sparse Gaussian Processes. In *Advances in Neural Information Processing Systems 28*, Montreal, Canada.

Hensman, J., Matthews, A. G. d. G., and Ghahramani, Z. (2015b). Scalable Variational Gaussian Process Classification. In *18th International Conference on Artificial Intelligence and Statistics*, pages 351–360, San Diego, California, USA.

Hjort, N., Holmes, C., Müller, P., and Walker, S. (2010). *Bayesian Nonparametrics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

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

Hunt, B. R., Sauer, T., James, and Yorke, A. (1992). Prevalence: A translation-invariant almost every on infinite-dimensional spaces. *Bulletin of the Amer. Math. Soc*, pages 217–238.

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

Kolmogorov, A. N., Chiriaev, A. N., and Lindquist, G., editors (1992). *Selected works of A. N. Kolmogorov. Volume II. , Probability theory and mathematical statistics*. Kluwer Academic Publishers.

Kuss, M. and Rasmussen, C. (2005). Assessing approximate inference for binary gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704.

Lawrence, N. (2003). Gaussian Process Latent Variable Models for visualisation of high dimensional data. In *In NIPS*.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast Sparse Gaussian process methods: The Informative Vector Machine. In *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press.

LeCun, Y. (2016). The MNIST database. http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Lloyd, C., Gunter, T., Osborne, M., and Roberts, S. (2015). Variational Inference for Gaussian Process Modulated Poisson Processes. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1814–1822.

MacKay, D. J. (1997). Ensemble learning for hidden markov models. Technical report, University of Cambridge.

MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

Maclaurin, D., Duvenaud, D., and Johnson, M. (2016). Autograd. *https://github.com/HIPS/autograd*.

Marimont, R. B. and Shapiro, M. B. (1979). Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Minka, T. (2001). Expectation propagation for approximate bayesian inference. In *Conference on Uncertainty in Artificial Intelligence*, pages 362–369.

Møller, J., Syversveen, A. R., and Waagepetersen, R. P. (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press.

Murray, I. (2016a). *https://github.com/imurray/chol-rev*.

Murray, I. (2016b). Differentiation of the Cholesky decomposition. *arXiv preprint 1602.07527*.

Murray, I., Adams, R. P., and MacKay, D. J. (2010). Elliptical slice sampling. *JMLR: W&CP*, 9:541–548.

Naish-Guzman, A. and Holden, S. (2008). The generalized FITC approximation. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 1057–1064. MIT Press, Cambridge, MA.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Lecture notes in statistics. Springer, New York, Berlin, Paris.

Neal, R. M. (2010). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078.

Opper, M. and Archambeau, C. (2009). The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792.

Orbanz, P. and Teh, Y. W. (2010). Bayesian Nonparametric Models. In *Encyclopedia of Machine Learning*. Springer.

Pinski, F., Simpson, G., Stuart, A., and Weber, H. (2015a). Algorithms for Kullback-Leibler approximation for probability measures in infinite dimensions. *SIAM Journal on Scientific Computing*, 37:A2733–A2757.

Pinski, F., Simpson, G., Stuart, A., and Weber, H. (2015b). Kullback-Leibler approximation for probability measures on infinite dimensional spaces. *SIAM J. Mathematical Analysis*, 47:4091–4122.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959.

Rao, D. (2016). The unreasonable popularity of tensorflow. http://deliprao.com/archives/168.

Rasmussen, C. E. and Nickisch, H. (2010). Gaussian Processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for machine learning*. The MIT Press.

Rätsch, G., Onoda, T., and Müller, K.-R. (2001). Soft margins for AdaBoost. *Mach. Learn.*, 42(3):287–320.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Saul, A. D., Hensman, J., Vehtari, A., and Lawrence., N. D. (2016). Chained Gaussian processes. In *19th International Conference on Artificial Intelligence and Statistics*, Cadiz, Spain.

Schervish, M. (1995). *Theory of Statistics*. Springer Series in Statistics. Springer.

Seeger, M. (2003a). *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh.

Seeger, M. (2003b). PAC-Bayesian generalisation error bounds for Gaussian process classification. *J. Mach. Learn. Res.*, 3:233–269.

Sengupta, A. N. (2014). The Kolmogorov extension theorem.

Smith, S. P. (1995). Differentiation of the Cholesky algorithm. *J. Comp. Graph. Stat.*, 4(2):134–147.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979. JMLR Workshop and Conference Proceedings.

Titsias, M. K. (2009a). Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 567–574.

Titsias, M. K. (2009b). Variational Model Selection for Sparse Gaussian Process regression. Technical report.

Titsias, M. K., Lawrence, N., and Rattray, M. (2011). Markov chain Monte Carlo algorithms for Gaussian processes. In Barber, D., Chiappa, A. T., and Cemgil, S., editors, *Bayesian time series models*.

Titsias, M. K. and Lawrence, N. D. (2010). Bayesian Gaussian process latent variable model. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*.

Turner, R. E. and Sahani, M. (2011). Probabilistic amplitude and frequency demodulation. In *Advances in Neural Information Processing Systems 24*, pages 981–989. The MIT Press.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., and Vehtari, A. (2013). GPstuff: Bayesian modeling with Gaussian processes. *J. Mach. Learn. Res.*, 14(1):1175–1179.

Wang, Z., Mohamed, S., and De Freitas, N. (2013). Adaptive Hamiltonian and Riemann manifold Monte Carlo. In *ICML*, volume 28, pages 1462–1470.

Wiener, N. and Masani, P. (1976). *Collected Works with Commentaries*, volume 1. MIT Press.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint 1212.5701*.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

Appendix A

Extending measures beyond the product σ-algebra

In section 1.1.4 we established that the product σ-algebra is missing some events that are potentially of interest in probabilistic models. In this section we outline some of the technical issues that surround consistently extending measures to a larger σ-algebra.

One difficulty is that the number of potential extensions is infinite and can give qualitatively difference answers as we shall now show through some examples. Firstly we exhibit a simple example due to Adler (1981) that demonstrates that a random function is not fully specified by its finite dimensional marginals outside the product σ-algebra. Let $X = [0, 1]$ and define two random functions f and g. All randomness will come from a single uniform random variate γ:

$$\gamma \sim \text{Uniform}[0, 1]\,. \tag{A.1}$$

f will be defined as:

$$f(x, \gamma) = 0 \quad \forall x, \gamma\,. \tag{A.2}$$

g will be defined as:

$$g(x, \gamma) = \begin{cases} 0, & \text{if } x \neq \gamma \\ 1, & \text{if } x = \gamma\,. \end{cases} \tag{A.3}$$

These two random functions have the same finite dimensional marginals. Obviously the finite dimensional marginals of f allocate all their probability to $f(t_i) = 0 \;\; i = 1, ..., N$. This is also true for g because for any such finite index set subset $\{x_1, ..., x_N\}$ of $[0, 1]$ the probability γ will land in this subset is 0. Thus the finite dimensional marginals for g allocate all their probability to $g(t_i) = 0 \;\; i = 1, ..., N$. But compare the event

$$E_c = \{\gamma : \text{function is continuous } \forall x \in [0, 1] \,\} \tag{A.4}$$

which has probability 1 for f and 0 for g.

The second example builds on the first example and is adapted from Billingsley (1995). Consider the sample functions in figure 1.2. The plots were actually generated using a finite mesh of test points which is then connected by the plotting tool. Intuition might suggest that this process necessarily somehow corresponds in the limit of a very fine mesh to some continuous underlying process. Here we will show that this intuition is wrong. Whilst there is a continuous *version* of the process in both cases this is not a necessary consequence of the finite dimensional marginals. Intuition can be misleading in the theory of stochastic processes.

Consider the *Dirichlet function* which is the indicator function of the rationals $\mathbb{1}_{\mathbb{Q}} : \mathbb{R} \mapsto [0, 1]$. This function has the property that it is discontinuous everywhere. Now suppose that we have a random function $f(x, \omega)$ where we have made explicit the random variable aspect of the function by showing the dependence on ω. Let us assume that this function is everywhere continuous as a function of x almost surely. Now again take $\gamma \sim \text{Uniform}[0, 1]$ and then define:

$$g(x, \omega, \gamma) = f(x, \omega) + \mathbb{1}_{\mathbb{Q}}(x + \gamma) \,. \tag{A.5}$$

As in the previous example, the function g has the same finite dimensional marginals as f because the probability that $x + \gamma$ is rational is zero. But we now have a random function g that is everywhere discontinuous almost surely.

When are these issues encountered? The intuition for the dividing line is that questions involving finite and countable sets are addressed by the product σ-algebra where intrinsically uncountable concepts like continuity, differentiability and boundedness need an extension. The theory of extensions beyond the product σ-algebra in the case where the index set has a dense countable subset (for example the rationals are a dense subset of the reals) was addressed by Doob (1953). Essentially the idea is to *assume* that the behaviour on the dense subset countable subset fully specifies the behaviour of the process. The process is then said to be *separable*.

Appendix B

Augmented models do not necessarily have the same variational optimum as unaugmented ones

In this section we prove theorem 4.(ii) numerically. It is sufficient to give one example where the augmented and unaugmented $\mathcal{KL}$-divergence have different optima. Such an example was not difficult to find. We take the special case discussed in proposition 2 where $X = D$ and $I = Z$ and assume all relevant densities exist. We take the likelihood to be Gaussian which means, as Titsias showed (2009a), that his target $\mathcal{KL}$-divergence has an analytic solution $q^*(f_Z)$ for the optimal inducing output distribution . We study the case of one inducing point and three randomly generated data points. The kernel variance, kernel lengthscale and noise variance are all fixed at 1.

If the divergences $\mathcal{KL}[q(f_{Z \cup D})||p(f_{Z \cup D}|Y)]$ and $\mathcal{KL}[q(f_D)||p(f_D|Y)]$ have the same optima for all variational parameters, then substituting $q^*(f_Z)$ into both $\mathcal{KL}$-divergences should give collapsed objectives that have the same optima for the one inducing input position. We plot the results of such an experiment in figure B.1. As is required by theory, the unaugmented $\mathcal{KL}$-divergence lower bounds the augmented one. Clearly the functions have different minima so we have our counter example and theorem 4.(ii) is proved.

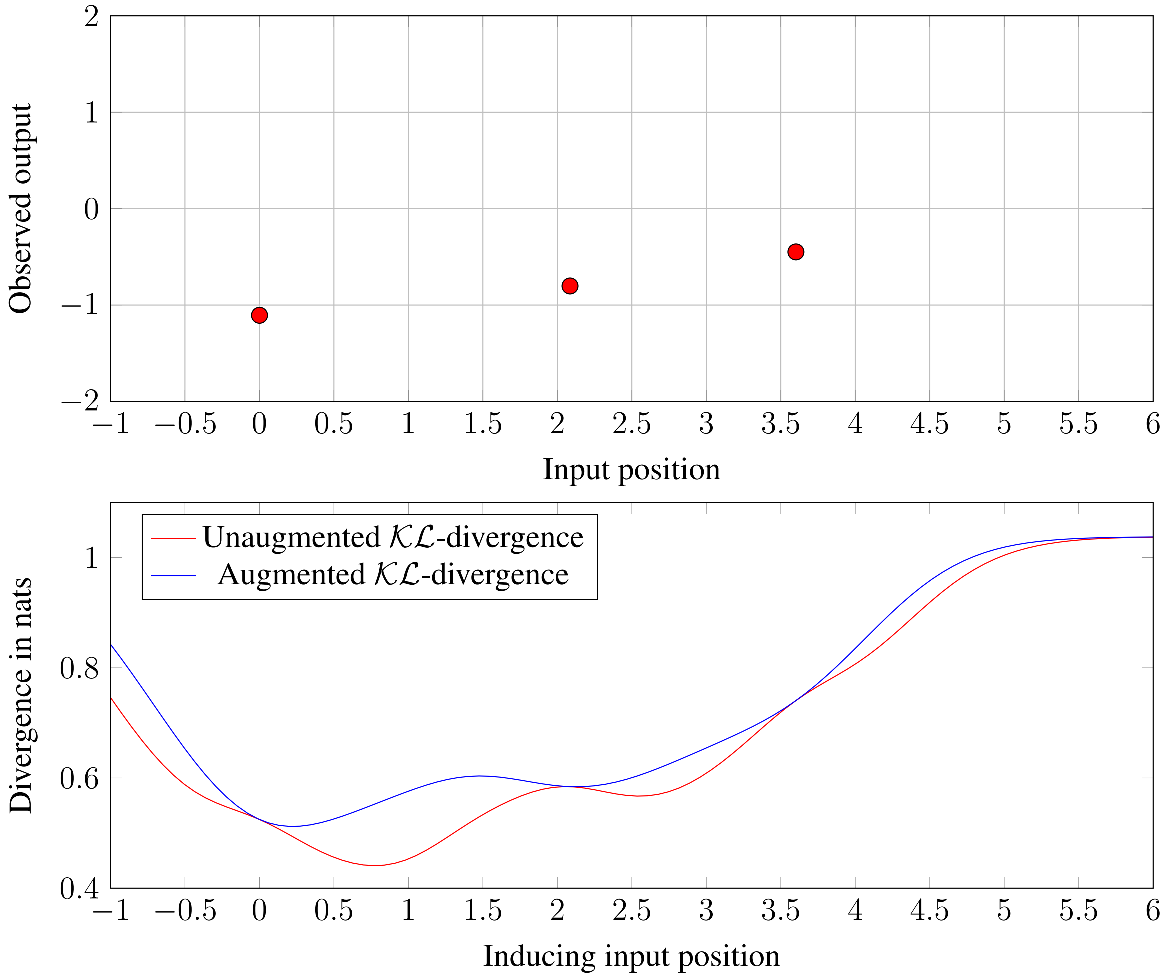

Fig. B.1 Comparing an augmented $\mathcal{KL}$-divergence to an augmented one on a simple randomly generated data set. Top: the randomly generated regression data. Bottom: a comparison of the $\mathcal{KL}$-divergence as a function of the single inducing input value. Clearly the two functions have different minima.

Appendix C

The equivalence between the inducing random variables framework and conditionally deterministic augmentation

Here we show that the variational inducing random variables approximation of section 3.5 corresponds to the augmentation framework of section 3.4 when the augmentation is conditionally deterministic. Let the measurable deterministic transformation in question be given by h_θ. Throughout this section $A_I \subset \Omega_I$ and $A_X \subset \Omega_X$ will be measurable sets. A conditionally deterministic model augmentation has the property that:

$$P_I(A_I) = P_X(h_\theta^{-1}(A_I)), \tag{C.1}$$

from which it follows that the joint distribution of f_I and f_X has the property

$$P_{X \cup I}(A_X \times \Omega_I) = P_I(h_\theta(A_X))\,. \tag{C.2}$$

In the augmentation framework, the way to define the approximating measure is given in equation (3.32), which we repeat here suppressing some parameter dependencies:

$$\frac{dQ_{X \cup I}}{dP_{X \cup I}}(f_{X \cup I}) = \frac{dQ_I}{dP_I}(\pi_I(f_{X \cup I}))\,. \tag{C.3}$$

Consider the marginal distribution Q_X of this approximating measure on the original index set X. From (C.3) it has the property that

$$Q_X(A_X) := Q_{X \cup I}(A_X \times \Omega_I) = \int_{A_X \times \Omega_I} \frac{dQ_I}{dP_I}(\pi_I(f_{X \cup I}))dP_{X \cup I}(f_{X \cup I})\,. \tag{C.4}$$

Let $C = \{f_{X \cup I} \in A_X \times \Omega_I : f_I = h_\theta(f_X)\}$. The complement of this set has measure zero under $P_{X \cup I}$. Therefore we have:

$$Q_X(A_X) = \int_C \frac{dQ_I}{dP_I}(\pi_I(f_{X \cup I}))dP_{X \cup I}(f_{X \cup I}) \tag{C.5}$$

$$= \int_C \frac{dQ_I}{dP_I}(h_\theta(f_X))dP_{X \cup I}(f_{X \cup I}) \tag{C.6}$$

$$= \int_{A_X \times \Omega_I} \frac{dQ_I}{dP_I}(h_\theta(f_X))dP_{X \cup I}(f_{X \cup I})\,. \tag{C.7}$$

Now applying the marginalization property of the integral we have

$$Q_X(A_X) = \int_{A_X} \frac{dQ_I}{dP_I}(h_\theta(f_X))dP_X(f_X)\,. \tag{C.8}$$

This implies that $\frac{dQ_I}{dP_I}(h_\theta(f_X))$ is a version of the Radon-Nikodym derivative $\frac{dQ_X}{dP_X}$. But this is precisely the defining property of the variational inducing random variables approximation (3.38) with $I = R$, so the equivalence is proved.

Appendix D

Properties of the normal distribution

D.1 Marginal Gaussian distributions

This identity can be found for instance in Bishop (2006) section 2.3.3.

Consider two vector random variables x and y. Let x have the marginal distribution

$$p(x) = \mathcal{N}(x|\mu, \Lambda) \tag{D.1}$$

Let y have the conditional distribution

$$p(y|x) = \mathcal{N}(y|Ax + b, L) \tag{D.2}$$

Then y has a marginal normal distribution given by

$$p(y) = \mathcal{N}(y|A\mu + b, L + A\Lambda A^T) \tag{D.3}$$

D.2 Derivatives of Gaussian integrals

Consider a normally distributed random variable x.

$$p(x) = \mathcal{N}(x|\mu, \Lambda) \tag{D.4}$$

We are interested in the derivatives of the integral

$$I = \int_{\mathbb{R}} p(x) f(x) dx \tag{D.5}$$

with respect to the parameters of the normal distribution. These are given by

$$\frac{dI}{d\mu} = \int_{\mathbb{R}} p(x) \frac{\partial f(x)}{\partial x} dx \tag{D.6}$$

and

$$\frac{dI}{d\Lambda} = \frac{1}{2} \int_{\mathbb{R}} p(x) \frac{\partial^2 f(x)}{\partial x^2} dx \,. \tag{D.7}$$

These expressions obviously have a long history but their utility in variational inference was first noted by Opper and Archambeau Opper and Archambeau (2009).

Appendix E

The Jacobian of the whitening transformation

Working within the hierarchical latent Gaussian model implied by the log density in equation (5.24), we are interested in the Jacobian of the transformation

$$v = \Lambda^{-1}(\theta) f_Z \tag{E.1}$$

where we have used the Cholesky decomposition Λ of $K_{Z,Z}$ so that $\Lambda\Lambda^T = K_{Z,Z}$. We need to be slightly careful because v depends both on f_Z and also θ through the Cholesky decomposition of the kernel gram matrix. The old parameters are

$$\Phi_{\text{Old}} = \begin{pmatrix} f_Z \\ \theta \end{pmatrix} \tag{E.2}$$

The new parameters are

$$\Phi_{\text{New}} = \begin{pmatrix} v \\ \theta \end{pmatrix} \tag{E.3}$$

Define the change of variables by the vector valued function g with

$$\Phi_{\text{New}} = g(\Phi_{\text{Old}}) \tag{E.4}$$

We will be interested in the Jacobian J of this transformation which has block form

$$J(g) = \begin{pmatrix} \Lambda^{-1} & E \\ 0 & I \end{pmatrix} \tag{E.5}$$

The matrix E has a complex form in terms of the hyperparameter differential of the Cholesky matrix. Fortunately we do not need to evaluate this to find the Jacobian determiant since, using the block formula for determinants:

$$\det\{J(g)\} = \det\{\Lambda^{-1}(\theta)\}\det\{I - 0\Lambda E\} = \det\{\Lambda^{-1}(\theta)\} \tag{E.6}$$

as claimed in section 5.4.

Appendix F

A C++ and Eigen implementation of Cholesky backpropagation for TensorFlow

F.1 Comments on code

The code is a version of Murray's blocked algorithm (2016b). At the time of writing Murray himself has made implementations available in Python, MATLAB and FORTRAN (Murray, 2016a).

The pull request of the code to the main TensorFlow GitHub repository, which is managed by Google, was reviewed by Rasmus Munk Larsen. The review involved detailed comments about tighter integration with the TensorFlow architecture and improved the general readability of the code. It also required us to conform to the coding style required by the Google C++ standard (Google, 2016). There have been some relatively small changes to the code since which unfortunately for some reason have overwritten the full commit and author history in the version control system. The code included here is as it was when the pull request was accepted. TensorFlow makes heavy use of Eigen (Guennebaud et al., 2010) an open source linear algebra library. This library makes heavy use of templating. In particular it uses Expression Templates to intelligently remove unnecessary temporaries and to perform lazy evaluation where appropriate. The latter is a feature more commonly associated with functional programming languages and is impressive in a C++ library. Although all this comes at relatively little degradation to the interface that a coder sees, some care is required to preserve the best performance. For example, the use of Eigen::Ref on lines 32 and 33 allows us to avoid the creation of unnecessary temporaries when calling functions.

The standard way to test gradient code in TensorFlow is by finite differences. As part of the pull request we added this test functionality for the CholeskyGrad Op.

F.2 TensorFlow source code

```
/* Copyright 2015 Google Inc. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#include "tensorflow/core/framework/op.h"
#include "third_party/eigen3/Eigen/Core"

#include "tensorflow/core/framework/op_kernel.h"

#include "tensorflow/core/kernels/linalg_ops_common.h"
#include "third_party/eigen3/unsupported/Eigen/CXX11/Tensor"
#include "tensorflow/core/framework/tensor_types.h"
#include "tensorflow/core/framework/types.h"

namespace tensorflow {

template <typename T> class CholeskyGrad : public OpKernel {
 public:
  explicit CholeskyGrad(OpKernelConstruction* context) : OpKernel(context) {}
  using Matrix =
      Eigen::Matrix<T, Eigen::Dynamic, Eigen::Dynamic, Eigen::RowMajor>;
  using ConstMatrixMap = Eigen::Map<const Matrix>;
  using MatrixMap = Eigen::Map<Matrix>;
  using ConstRef = Eigen::Ref<const Matrix>;
  using Ref = Eigen::Ref<Matrix>;

  void Compute(OpKernelContext* context) override {
    const Tensor& input_tensor_l = context->input(0);
    const Tensor& input_tensor_grad = context->input(1);
    // Check that input tensors represent a matrix.
    OP_REQUIRES(context, TensorShapeUtils::IsMatrix(input_tensor_l.shape()),
                errors::InvalidArgument("In[0] is not a matrix"));
    OP_REQUIRES(context, TensorShapeUtils::IsMatrix(input_tensor_grad.shape()),
                errors::InvalidArgument("In[1] is not a matrix"));
    // Check that input tensors are square.
    OP_REQUIRES(context,
                input_tensor_l.dim_size(0) == input_tensor_l.dim_size(1),
                errors::InvalidArgument("Input matrix must be square."));
    OP_REQUIRES(context,
                input_tensor_grad.dim_size(0) == input_tensor_grad.dim_size(1),
                errors::InvalidArgument("Input matrix must be square."));

    // Check that input tensors are of same size.
    OP_REQUIRES(context,
                input_tensor_l.dim_size(0) == input_tensor_grad.dim_size(0),
                errors::InvalidArgument("Input matrices must be same size."));

    // Create an output tensor
    Tensor* output_tensor = NULL;
    OP_REQUIRES_OK(context, context->allocate_output(
                                0, input_tensor_grad.shape(), &output_tensor));

    if (output_tensor->NumElements() == 0) {
      // the output shape is a 0-element matrix, so there is nothing to do.
      return;
    }
    // The next lines are necessary to get Eigen matrix behaviour.
    const ConstMatrixMap input_matrix_l_full(input_tensor_l.flat<T>().data(),
                                             input_tensor_l.dim_size(0),
                                             input_tensor_l.dim_size(1));
    const ConstMatrixMap input_matrix_grad(input_tensor_grad.flat<T>().data(),
                                           input_tensor_grad.dim_size(0),
                                           input_tensor_grad.dim_size(1));
    MatrixMap output_matrix(output_tensor->template flat<T>().data(),
                            input_tensor_l.dim_size(0),
                            input_tensor_l.dim_size(1));

    // Algorithm only depends on lower triangular half on input_tensor_l.
    const Matrix input_matrix_l =
        input_matrix_l_full.template triangularView<Eigen::Lower>();
    // Algorithm only depends on lower triangular half on input_matrix_grad.
    output_matrix = input_matrix_grad.template triangularView<Eigen::Lower>();

    const int64 kMatrixSize = input_matrix_l.rows();
    const int64 kMaxBlockSize = 32;

    for (int64 block_end = kMatrixSize; block_end > 0;
         block_end -= kMaxBlockSize) {
```

```
      /* This shows the block structure.

      /      \
      |      |
      | R D  |
      \ B C  /

      Variables names representing the derivative matrix have a trailing '_bar'.
      */

      const int64 block_begin = std::max(0ll, block_end - kMaxBlockSize);
      const int64 block_size = block_end - block_begin;
      const int64 trailing_size = kMatrixSize - block_end;

      auto B = input_matrix_l.block(block_end, 0, trailing_size, block_begin);
      auto B_bar =
          output_matrix.block(block_end, 0, trailing_size, block_begin);

      auto C = input_matrix_l.block(block_end, block_begin, trailing_size,
                                    block_size);
      auto C_bar = output_matrix.block(block_end, block_begin, trailing_size,
                                       block_size);

      auto D = input_matrix_l.block(block_begin, block_begin, block_size,
                                    block_size);
      auto D_bar =
          output_matrix.block(block_begin, block_begin, block_size, block_size);

      auto R = input_matrix_l.block(block_begin, 0, block_size, block_begin);
      auto R_bar = output_matrix.block(block_begin, 0, block_size, block_begin);

      C_bar = D.adjoint().template triangularView<Eigen::Upper>()
          .solve(C_bar.adjoint()).adjoint();
      D_bar -= (C_bar.adjoint() * C).template triangularView<Eigen::Lower>();
      B_bar -= C_bar * R;
      R_bar -= C_bar.adjoint() * B;
      CholeskyGradUnblocked(D, D_bar);
      R_bar -= (D_bar + D_bar.adjoint()) * R;
    }
    output_matrix = (0.5 * (output_matrix + output_matrix.transpose())).eval();
  }
  void CholeskyGradUnblocked(const ConstRef l_block, Ref grad_block) {
    const int64 kMatrixSize = l_block.rows();
    for (int64 k = kMatrixSize - 1; k >= 0; k--) {
      /* This shows the block structure.

      /       \
      |       |
      | r d   |
      \ B c   /

      Variables names representing the derivative matrix have a trailing '_bar'.
      */

      const int64 number_rows_B = kMatrixSize - (k + 1);
      const int64 number_rows_r_stack_B = number_rows_B + 1;

      auto r = l_block.block(k, 0, 1, k);
      auto r_bar = grad_block.block(k, 0, 1, k);
      auto d = l_block(k, k);  // This needs to be a scalar rather than a view.
      auto d_bar = grad_block.block(k, k, 1, 1);
      // B is not included explicitly because it is not used on its own.
      auto B_bar = grad_block.block(k + 1, 0, number_rows_B, k);
      auto c = l_block.block(k + 1, k, number_rows_B, 1);
      auto c_bar = grad_block.block(k + 1, k, number_rows_B, 1);
      // Result of vertical stacking d_bar and c_bar.
      auto d_stack_c_bar = grad_block.block(k, k, number_rows_r_stack_B, 1);
      // Result of vertical stacking of r and B.
      auto r_stack_B = l_block.block(k, 0, number_rows_r_stack_B, k);
      d_bar -= (c.adjoint() * c_bar) / d;
      d_stack_c_bar /= d;
      r_bar -= d_stack_c_bar.adjoint() * r_stack_B;
      B_bar -= c_bar * r;
      d_bar /= 2.;
    }
  }
};

REGISTER_LINALG_OP("CholeskyGrad", (CholeskyGrad<float>), float);
REGISTER_LINALG_OP("CholeskyGrad", (CholeskyGrad<double>), double);
}  // namespace tensorflow
```