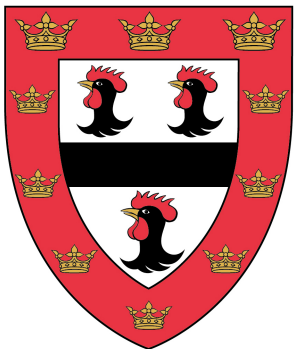
Sparse Gaussian Process Approximations and Applications



Mark van der Wilk

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College

November 2018

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.

Mark van der Wilk
November 2018

Acknowledgements

The journey towards finishing my PhD was influenced (and improved) by a considerable number of people. To start, I would like to thank my supervisor, Carl Rasmussen for all the insight and advice. I always enjoyed his clarity of thought, particularly the many explanations which hit the nail on the head in a minimal (and sometimes entertaining) way. I also very much appreciated working with James Hensman towards the end of my PhD. I learned a great deal from his vast knowledge on how to manipulate Gaussian processes, and from his optimism and knack for identifying where our models would be most practically useful.

I would also like to thank Matthias Bauer for a fruitful collaboration, and members of the research group for continuously useful, enjoyable and challenging discussions, particularly Koa Heaukulani, Yarin Gal, David Lopez-Paz, Rowan McAllister, Alessandro Ialongo, David Duvenaud, and Roger Frigola. The stimulating environment in the Machine Learning Group would, of course, not have been possible without Zoubin Ghahramani, Carl Rasmussen, and Richard Turner. I would also like Richard Turner and Neil Lawrence for examining my thesis and the enjoyable discussions during the viva, and their suggestions for improvement.

This PhD was funded by the UK Engineering and Physics Research Council (EPSRC), and Qualcomm, through the Qualcomm Innovation Fellowship (2015).

On a personal note, thanks to Rainbow for her patience and brightness throughout the entire journey. And finally, I would like to thank my parents, Nino and Katy, for spurring on my interest in science by showing me how things work, and for teaching me that great outcomes are accompanied by great effort.

Abstract

Many tasks in machine learning require learning some kind of input-output relation (function), for example, recognising handwritten digits (from image to number) or learning the motion behaviour of a dynamical system like a pendulum (from positions and velocities now to future positions and velocities). We consider this problem using the Bayesian framework, where we use probability distributions to represent the state of uncertainty that a learning agent is in. In particular, we will investigate methods which use Gaussian processes to represent distributions over functions.

Gaussian process models require approximations in order to be practically useful. This thesis focuses on understanding existing approximations and investigating new ones tailored to specific applications. We advance the understanding of existing techniques first through a thorough review. We propose desiderata for non-parametric basis function model approximations, which we use to assess the existing approximations. Following this, we perform an in-depth empirical investigation of two popular approximations (VFE and FITC). Based on the insights gained, we propose a new *inter-domain* Gaussian process approximation, which can be used to increase the sparsity of the approximation, in comparison to regular inducing point approximations. This allows GP models to be stored and communicated more compactly. Next, we show that inter-domain approximations can also allow the use of models which would otherwise be impractical, as opposed to improving existing approximations. We introduce an inter-domain approximation for the Convolutional Gaussian process – a model that makes Gaussian processes suitable to image inputs, and which has strong relations to convolutional neural networks. This same technique is valuable for approximating Gaussian processes with more general invariance properties. Finally, we revisit the derivation of the Gaussian process State Space Model, and discuss some subtleties relating to their approximation.

We hope that this thesis illustrates some benefits of non-parametric models and their approximation in a non-parametric fashion, and that it provides models and approximations that prove to be useful for the development of more complex and performant models in the future.

Table of contents

List of figures

List of tables

Nomenclature

Symbols

$\mathbf{f}$	Shorthand for $f(X)$ if there is no ambiguity.
$f(\mathbf{x})$	The evaluation of $f(\cdot)$ at a single location $\mathbf{x}$.
$f(X)$	$f([X]_n) = f(\mathbf{x}_n)$, i.e. the evaluation of $f(\cdot)$ at each location in X, where $X \in \mathcal{X}^N$ and $f : \mathcal{X} \to \mathcal{Y}$. Usually $X \in \mathbb{R}^{N \times D}$ and $f(X) \in \mathbb{R}^N$.
$\mathcal{X}$	The space of inputs to a particular function.
M	Number of inducing points.
X	Matrix of inputs $\in \mathbb{R}^{N \times D}$. Each input is $\in \mathbb{R}^D$.
$\mathbf{y}$	Vector of observations $\in \mathbb{R}^N$.
$\mathcal{L}$	Variational lower bound on the LML.
N	Number of observations / data points.
$\mathcal{Y}$	The space of outputs from a particular function.

Acronyms / Abbreviations

DTC	Deterministic Training Conditional
ELBO	Evidence Lower BOund – the variational bound on the marginal likelihood, or the negative Variational Free Energy (VFE).
EP	Expectation Propagation
FIF	Frequency inducing feature
FITC	Fully Independent Training Conditional

GP Gaussian process

GPLVM Gaussian Process Latent Variable Model

GPSSM Gaussian Process State Space Model

KL Kullback-Leibler, usually in the context of the divergence.

LML Log marginal likelihood

MSIF Multi-scale inducing feature

NLML Negative log marginal likelihood

RFF Random Fourier Feature

SE Squared Exponential (type of kernel)

TFIF Time-frequency inducing feature

VFE Variational Free Energy - The negative lower bound on the LML of the full GP.

VI Variational inference

Notation

While the Gaussian processes discussed in this thesis can often have very general input and output spaces, we mainly consider real valued inputs and outputs in this thesis. While doing so, we manipulate the data to various degrees of granularity. We shall take the following convention:

- Inputs are considered to be $\in \mathbb{R}^D$, while outputs are considered to be univariate ($\in \mathbb{R}$).
- A collection of N inputs is denoted as a matrix $X \in \mathbb{R}^{N \times D}$.
- The nth input vector in X is denoted as $\mathbf{x}_n$.
- A specific dimension will be denoted using the subscript d, i.e. x_{nd} for the dth dimension of the nth input, or x_d for the dth dimension of a general input $\mathbf{x}$.
- Outputs for multiple inputs will generally be considered vectors $\in \mathbb{R}^N$.
- We denote an entire function as $f(\cdot)$, and a particular evaluation at $\mathbf{x}$ as $f(\mathbf{x})$.
- Multiple evaluations can be written as $f(X) \in \mathbb{R}^N$, where X contains N inputs.

Chapter 1

Introduction

Machine learning aims to create algorithms that improve their performance by leveraging observed data or experience on a task. For example, we may want a computer to use examples of hand-written digits to learn how to recognise them. Generally, machine learning algorithms rely on statistical models which are built to reflect reality (or at least be useful for predicting it). Statistical models may contain parameters which are unknown or uncertain *a-priori* (i.e. when starting out before observing any data, or gaining any experience). Given an increasing amount of data, learning involves a) finding parameter settings which will improve the performance of the task, and b) choosing which statistical models are likely to perform well.

In essence, this is identical to familiar problems in statistics, e.g. where we may want to predict the risk of a disease using certain environmental factors. Generally, the approach is to propose multiple models (e.g. linear and non-linear) which relate the observed factors and the presence of disease. As more data is gathered and made available to the model, the relationship between the observations becomes clearer, leading to more certain parameter estimates. Additionally, we can also find out which model explains the relationship best (e.g. whether a linear model suffices, or whether the more complicated non-linear one is needed). Both of these improvements allow us to make better predictions.

When following this procedure with any finite dataset, we can not expect to get completely certain answers. Each model may contain multiple different parameter settings which fit the data equally well, but which may give different predictions for unseen inputs. In fact, the parameter which best fits the training data, may predict very poorly in the future. This is known as *overfitting*, and must be avoided in some way by any machine learning technique.

This chapter provides an introduction to the Bayesian approach to machine learning and Gaussian processes as Bayesian models. The main aims are to describe fundamental Bayesian and Gaussian process methodologies, and to emphasise their elegant properties which make them worthy to study.

1.1 Bayesian Machine Learning

The two problems described above (finding parameters in models, and assessing the models themselves) can be neatly solved within the single framework of Bayesian inference[1]. To illustrate the problem further, figure 1.1 shows an example regression task with three possible models to explain the relationship. Each model contains parameters which determine the shape of the regression line. A single setting for the parameters corresponds to a fully defined regression line. Given the limited dataset, there are many parameter settings in each model which are consistent with the data. Additionally, it is also unclear whether the relationship is linear or quadratic – both fit the data to a degree. It should be clear, however, that the high order polynomial is probably not going to give sensible predictions, despite it having the smallest deviations from the training data.

The polynomial solution is a classic example of overfitting, and shows that a solution which fits the training data may not generalise to future problems. The missing ingredient is a way to limit the *complexity* of the solution in the right way. In the Bayesian framework, complexity is automatically managed by correctly keeping track of the *uncertainty* around the quantity of interest [Rasmussen and Ghahramani, 2001]. Other machine learning frameworks have different methods for dealing with complexity (see e.g. Mohri et al. [2012, ch.3] or Grünwald [2007]).

1.1.1 Probability theory: a way to reason under uncertainty

Figure 1.1 illustrates that there are many different solutions to a regression problem that all seem plausible to some degree. Taking again the example of predicting disease, where the x-axis would be a risk factor (like eating ice-cream) and the y-axis the risk of getting ill, the current data cannot unambiguously determine the truth of a statement like "halving my intake of ice-cream will reduce my risk of disease by 50%". Things become more complicated with added levels of uncertainty. What if the relationship only existed in people with a particular gene, and I'm uncertain about whether I have

[1] See Ghahramani [2012] or Ghahramani [2015] for good introductions and MacKay [2002] or Jaynes [2003] for an extensive treatment.

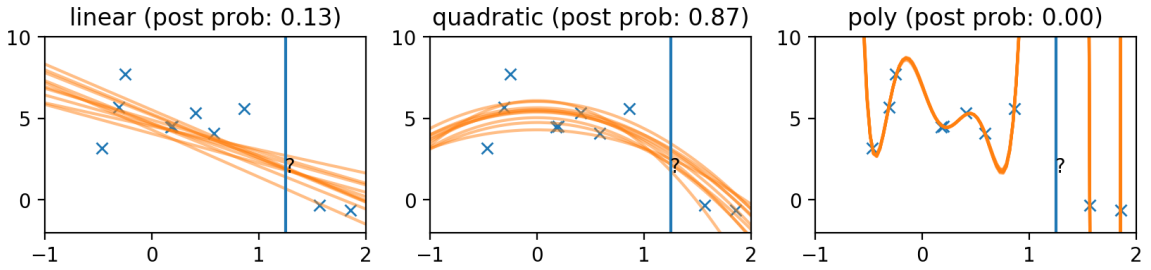


Fig. 1.1 Example regression problem with linear and quadratic models using Bayesian inference, and an 11th order polynomial model chosen to minimise the error to the training points for comparison. We want to predict at the point marked '?'. Plausible regression functions are sampled from the models' posteriors, while optimising for data fit seems to lead to nonsensical (and off-the-scale) predictions.

it? It would be nice if we had some kind of system for calculating our "degree of belief" for new statements, given how certain we are about the statements it depends on. Cox [1946] provided just this[2], by stating certain quantitative relations that such a calculus would need to obey in order to [Jaynes, 2003]:

- represent beliefs using real numbers,
- fit with common sense (i.e. beliefs in statements and their negations must vary in opposite directions with new information),
- be consistent (i.e. agents with the same starting beliefs must obtain the same results).

Cox [1946] showed that probability theory provides such a calculus, meaning we can represent beliefs about statements using probabilities.

1.1.2 Bayesian modelling & inference

Following the notion that we can represent uncertainty about quantities using probability distributions, we can formulate models about how the world works in terms of random variables. A model will generally make a prediction based on some configuration of the world. In mechanics, for example, Newton's laws of motion will make very precise predictions about the motion of objects, given perfect knowledge of the objects' positions and speeds[3]. Generally, we don't think about these predictions as

[2]Good discussions are also found in Jaynes [2003, ch 1 & 2] and MacKay [2002].

[3]Here, "precise predictions" refers to the model's claim that the prediction is precise. Newton's laws make precise predictions that are also correct (to some accuracy), which makes it a great model

being in terms of a probability distribution, but we don't expect the measurements to be *exactly* correct either. Phrasing the expected precision as a probability distribution is a natural way of expressing the remaining uncertainty. In probabilistic modelling, the distribution characterising the prediction given "all knowledge of the state of the world" is termed the *likelihood*. Uncertainty in the likelihood often represents the limit of the predictive capability of a model, beyond which the model can not be more precise. This can be the result of a physical limitation, or a deficiency in the model. For example, Newtonian mechanics is not accurate at relativistic speeds (a model deficiency), and the precise timing of the radioactive decay of a specific particle cannot be predicted (a physical limitation – as far as our models know...). These sources of inaccuracy or uncertainty can be accounted for in the likelihood.

Equally as important to the likelihood is our belief about the state of the world, which most of the time is not fully observed and uncertain. In probabilistic modelling the state of the world is expressed as a collection of *latent variables* or *parameters*. Again, taking mechanics as an example, we may still want to make predictions based on Newton's laws of motion if we are uncertain about the starting position and velocity of an object. The uncertainty in our prediction should combine the uncertainty of our state of the world, together with that of the likelihood.

Machine learning tasks can often be phrased as first reducing uncertainty about latent variables from data, and then making predictions based on this reduced uncertainty. Keeping with mechanics as an example, we can imagine needing to predict the orbit of an asteroid based on telescope measurements over several days. Given the limited precision of our measurements, we will have some residual uncertainty of the location and speed of the asteroid, which needs to be taken into account.

The process of going from data to conclusions about the latent variables is called *inference*[4], and is the main reason that allows machine learning models to improve their performance with more data. The more data, the more certain we can be about our latent variables, and the better we can predict. To a certain extent, inference is synonymous with learning. From the Bayesian point of view, the processes of inference and prediction require only manipulating the random variables and their probability distributions[5].

in many situations. A bad model makes precise predictions which are wrong. The theory of Brownian motion does not make precise predictions about the location of a specific particle, but nevertheless accurately predicts other properties.

[4]Or "statistical inference", to emphasise the presence of uncertainty and distinguish it from inference in pure logic.

[5]Here we start to take mathematical background for granted. MacKay [2002, ch.2] provides an excellent introduction to the rules of probability and manipulating distributions.

Very complicated models are possible with hierarchies of latent variables, but here we stick with the example in figure 1.1, as it contains the two main characteristics we are interested in: First, inferring the probability distribution over parameters (latent variables) in a particular model, and second, finding how certain to be in each model itself.

Parameter inference

The starting point in Bayesian inference is the *prior* distribution, which represents our uncertainty about parameters in the model *before* any data is observed (*a-priori*). We use our data and likelihood to update the prior distribution to represent our reduced uncertainty after taking the data into account (*a-posteriori*). In the regression problem above (and in this thesis) we can assume that we observe a function plus some Gaussian noise to account for any deviations. Taken together, the prior, likelihood and assumptions that went into choosing them make up our *model* and form a joint probability distribution over any combination of parameters and data that our model can capture:

$$\underbrace{p(\overbrace{\mathbf{w}}^{\text{parameter}}, \overbrace{\mathbf{y}}^{\text{data}})|\theta)}_{\text{joint under the model } \theta} = \underbrace{p(\mathbf{y}|\mathbf{w},\theta)}_{\text{likelihood}} \underbrace{p(\mathbf{w}|\theta)}_{\text{prior}} . \tag{1.1}$$

Once we observe some data, we are interested in the probability distribution of our parameters, given the observed data, the *posterior*. The distribution for this can be found using *Bayes' rule*:

$$p(\mathbf{w}|\mathbf{y},\theta) = \frac{p(\mathbf{y}|\mathbf{w},\theta)p(\mathbf{w}|\theta)}{p(\mathbf{y}|\theta)} = \frac{p(\mathbf{y}|\mathbf{w},\theta)p(\mathbf{w}|\theta)}{\int p(\mathbf{y}|\mathbf{w},\theta)p(\mathbf{w}|\theta)\mathrm{d}\mathbf{w}} . \tag{1.2}$$

The posterior completely captures our knowledge (and uncertainty) about the correct parameter for the model θ. Figure 1.1 visualises the uncertainty by taking samples from the posterior for each of the models.

Model comparison & marginal likelihoods

As said earlier, we are also uncertain about which of the two models is correct. Luckily, the correct model is just another unobserved variable that can be inferred from the data using Bayes' rule [MacKay, 2002, ch.28]. If we define the prior probabilities over the linear and quadratic as 0.5 each (so as not to favour one a-priori), we can again

use Bayes rule:

$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)\overbrace{p(\theta)}^{=0.5}}{p(\mathbf{y})}\,. \tag{1.3}$$

For θ being linear and quadratic in figure 1.1, the posterior probabilities end up being 0.13 and 0.87 respectively. Although the data prefers the quadratic model, we do still have some degree of belief for the linear one.

Interestingly, the likelihood for the model comparison task, is the normalising constant of the parameter estimation task. We call $p(\mathbf{y}|\theta)$ the *marginal likelihood*, and it has interesting properties that allow it to capture model complexity. We will review some of these characteristics more in section 5.1.1. For now, it suffices to note that it is really nice that the same procedure can be used for parameter inference and model inference. This is because the distinction between "parameters" and "model" is rather arbitrary. In the Bayesian framework, both are random variables which are to be inferred given some data. The structure of our inference problem can be much more complicated, with many levels in the hierarchy, all of which can be handled in the same way.

1.1.3 Predictions and decisions

So far, the Bayesian framework has shown us how to find the posterior distributions over parameters and models. When making predictions, we want to take our uncertainty into account. Predicting with a single parameter value, or single model would lead to over-confident results. Simply applying the same rules of probability will automatically take the uncertainty into account when finding the distribution over the prediction (y^*):

$$p(y^*|\mathbf{y}) = \int p(y^*|\mathbf{w},\theta)p(\mathbf{w}|\mathbf{y},\theta)p(\theta|\mathbf{y})\mathrm{d}\mathbf{w}\mathrm{d}\theta\,. \tag{1.4}$$

Based on the information given, we often need to choose some action, which will have varying outcomes, based on the actual outcome of the predicted quantity. Decision theory (see MacKay [2002, ch.36] or the relevant chapters in Rasmussen and Williams [2005]) tells us how to pick the optimal action. The predicted uncertainty can make a crucial difference in the optimal action to take. For example, imagine we are tying to throw balls into buckets, each of which gives a different score. The distribution of y^* describes where we think the ball will land relative to where we aim. If the distribution has low variance (i.e. we are good at throwing), it makes sense to aim at the highest

value bucket, even if there are low value buckets nearby. If the variance is high, we should find a region with many reasonably highly valued buckets to aim for.

The mathematical goal is easy to define. We simply have to define a loss function, which determines the loss for each pair of outcome y^* and action a. Then we minimise the expected value under our posterior predictive distribution:

$$a_{best} = \underset{a}{\operatorname{argmin}} \int L(\mathbf{y}^*, a) p(y^*|\mathbf{y}) \mathrm{d}y^* \,. \tag{1.5}$$

To what extent do we care about parameters?

Looking at the above criterion for decision making, we note that it only depends on the predictive distribution of the model (equation 1.4). If all we really care about is making predictions, and decisions based upon them, the central distribution of interest is not really the parameter posterior $p(\theta|\mathbf{y})$, but rather the predictive distribution $p(y^*|\mathbf{y})$, which can also be obtained from the marginal data distribution directly:

$$p(y^*, \mathbf{y}) = \int p(y^*|\mathbf{w}, \theta) p(\mathbf{y}|\mathbf{w}, \theta) p(\mathbf{w}|\theta) p(\theta) \mathrm{d}\mathbf{w} \mathrm{d}\theta \,, \tag{1.6}$$

$$p(y^*|\mathbf{y}) = \frac{p(y^*, \mathbf{y})}{p(\mathbf{y})} \,. \tag{1.7}$$

In this view, all distinctions between prior and likelihood vanish, and everything is phrased in terms of probability distributions on observable quantities. This is reassuring, as it shows that the result of Bayesian inference is invariant to the arbitrary choice we have in defining parameters, and decisions on whether to place effects in the prior or likelihood. The only thing that matters is the effect of choices on the data distribution.

So why do we care about the posterior over parameters, and why do we put so much emphasis on the distinction between prior and likelihood? There are two main reasons. Firstly, the separation between prior and likelihood can be mathematically convenient. Usually, parameters in the prior are chosen to make observations conditionally independent. This makes the posterior over parameters sufficient for making predictions in the future, which means we can forget about the details of the data we observed in any further analysis. This structure also is useful for making approximations. Secondly, we make these choices and distinctions for human convenience. Conditional independency is a useful way to isolate dependencies in models and are an intuitive way to describe structure in data. We may also want to interpret parameter values learned from data.

1.1.4 Practical considerations

While the Bayesian framework is clear in prescribing what computations need to be performed in order to solve a problem, it often prescribes computations which are far too costly to compute exactly – integrals being the main problem. In most real models, the posterior is not a distribution with a closed-form expression. Often the issue comes from the marginal likelihood being intractable [MacKay, 2002, ch.29], but it can also be worse. This problem has spurred on the development of many approximation methods like MCMC [Brooks et al., 2011], or variational methods [Beal, 2003; Blei et al., 2016].

One saving grace is that often when there are few parameters relative to the size of the dataset, the posterior will become very concentrated around one value. To illustrate this with an example, consider a model with i.i.d. observations and a continuous θ, where the posterior becomes:

$$p(\theta|\mathbf{y}) = \frac{\left[\prod_{n=1}^{N} p(y_n|\theta)\right] p(\theta)}{\int \left[\prod_{n=1}^{N} p(y_n|\theta)\right] p(\theta) \mathrm{d}\theta} \,. \tag{1.8}$$

Each data point adds a likelihood term which removes an area of density from the prior. The marginal likelihood normalises the model, causing the mass to concentrate around a point[6]. In situations like these the posterior can often be well approximated by a single point mass, obtained from maximising the posterior density. In practice, we will make this approximation when performing model comparison.

The models we consider in this work will have many (or rather, an infinite number of) parameters ($\mathbf{w}$ above), so we can not expect the posterior $p(\mathbf{w}|\mathbf{y}, \theta)$ to be peaked, making correct Bayesian inference important. However, we will only perform model comparison on a small number of models, or between models that are continuously parameterised by only a few parameters, so the posterior $p(\theta|\mathbf{y})$ is likely to be well-approximated by a point mass. The example in figure 1.1 is actually rather unusual in that the two models have a similar posterior probability. In many cases, the posterior probabilities quickly concentrate on a single model, so that only the most probable has any appreciable influence on the final predictions. This is discussed in detail in MacKay [1999] and Rasmussen and Williams [2005, §5.2].

[6] Assuming the likelihood doesn't contain equivalent values for different parameter settings.

1.2 Learning mappings with Gaussian processes

Like the regression problem in the previous section, many learning problems can be reduced to learning about a mapping from a space of inputs $\mathcal{X}$ to a space of outputs $\mathcal{Y}$. In classification $\mathcal{Y}$ is a set of discrete values (e.g. a number representing a digit), while in regression it is continuous (e.g. the state of a dynamical system in the future).

If we want to use the Bayesian framework, the mapping has to be represented as a random variable, with our current state of knowledge represented as its distribution. We start by first defining a prior distribution consistent with our beliefs, and then learn by updating our beliefs after observing data using Bayes' rule. Gaussian processes (GPs) are a class of distributions over functions which can be used for representing prior and posterior beliefs over mappings for the applications mentioned above [Rasmussen and Williams, 2005].

Over the next few sections we will review how to manipulate Gaussian process priors and posteriors, and relate them to other common machine learning models. A more in-depth discussion can be found in Rasmussen and Williams [2005]. In this discussion, however, we want to emphasise the connection to stochastic processes and finite basis function models. The stochastic process view in particular has recently gained traction after Matthews et al. [2016] showed it to be more clear and precise for describing approximate inference in Gaussian processes. Gaussian processes have been discussed in this way before in the context of machine learning, notably by Seeger [2003, 2004], although this generally requires more mathematical machinery (such as measure theory), making it less accessible. Here we aim to present without any measure theory, while still emphasising the stochastic process nature of the model.

1.2.1 Basis function models

In order to learn mappings with Bayesian inference, we must first consider how to place prior distributions on them. Functions can be represented conveniently as a sum of *basis functions*. We can define a family of functions by taking some simple functions as building blocks[7], and weighting them together:

$$f(x) = \sum_{m=1}^{M} w_m \phi_m(x)\,. \tag{1.9}$$

[7]For this example, we use "squared exponential" basis functions of the form $\phi_m = s^2 \exp\left(-\frac{1}{2}(x-c)^2\right)$. The chosen basis functions have a big influence on the properties of the resulting function.

Almost all statistical and machine learning models can be interpreted as representing functions this way, from linear models [Seal, 1967] and the earliest neural networks [Rosenblatt, 1958] to deep learning models [e.g. Krizhevsky et al., 2012] and Gaussian processes [Rasmussen and Williams, 2005]. Learning entails modifying the weights $\{w_i\}$ such that the resulting function fits the data. Figure 1.2 shows an example of basis functions being weighted to create some complex function.

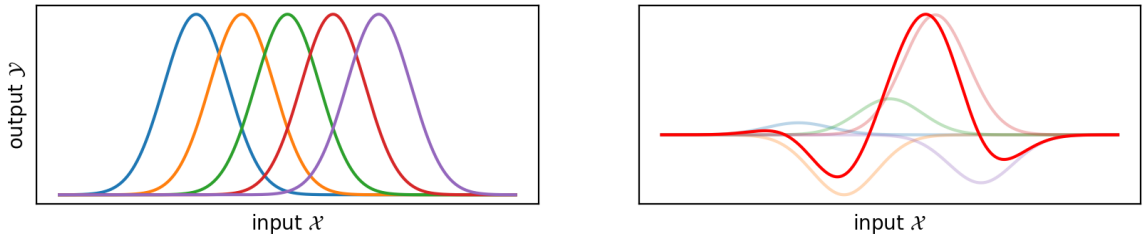


Fig. 1.2 Example of unweighted basis functions (left) and their combination into some more complex mapping from $\mathcal{X}$ to $\mathcal{Y}$ (right, combination in red, with its basis function components in the same colours as left).

Priors

Given the representation of functions in terms of basis functions, we can define a distribution over functions by placing a prior over the weights. We can equivalently view the weights $\{w_i\}$ or the entire function $f(\cdot)$ as a random variable, and we can obtain their posteriors through Bayes' rule. We can start with an independent Gaussian distribution for the weights:

$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}; 0, \sigma_w^2 I\right). \tag{1.10}$$

While we can easily represent distributions over a finite number of weights, it requires more careful measure theory to correctly handle random variables in the space of functions. However, it is important to note that the concept of the latent function as a random variable is a useful one. In this instance, manipulating the distribution on the weights is equivalent to manipulating the distribution over functions.

As discussed in section 1.1, the properties of the prior distribution need to match our prior beliefs about what functions we are likely to encounter. We will see more examples later, but in our current model we can adjust the properties of the functions by changing the basis functions or $p(\mathbf{w})$. Choosing a larger σ_w for example would place

higher probability on functions with large values. Viewing samples is a great way to convey the properties of functions that are likely under our chosen prior (figure 1.3).

It is also worth pointing out that when using a finite number of basis functions only a limited selection of functions can be represented. When using M basis functions, we can only freely specify the function value at M points, after which the weights are fully specified. All functions that are outside the span of our basis functions cannot be represented, and therefore are outside the prior. From equation 1.2 it should be clear that functions that are not in the prior, can also not be in the posterior, and can therefore not be learned. To guarantee a good performing model, we need to ensure that the prior is rich enough to contain functions (close to) the function we want to learn.

Posteriors

Next, we find the posterior of the weights after observing data. We assume our observations of the underlying function are corrupted by Gaussian noise. The likelihood can be formulated as either depending on the function value, or equivalently the weights:

$$p(y_n|f(x_n)) = \mathcal{N}\Big(y_n; f(x_n), \sigma^2\Big)\,, \tag{1.11}$$

$$p(y_n|\mathbf{w}, x_n) = \mathcal{N}\left(y_m; \sum_m w_m \phi_m(x_n), \sigma^2\right). \tag{1.12}$$

We find the posterior through Bayes' rule, and we can visualise samples in figure 1.3.

$$p(\mathbf{w}|\mathbf{y}) = \frac{\prod_n p(y_n|\mathbf{w}, x_n) p(\mathbf{w})}{p(\mathbf{y})}\,. \tag{1.13}$$

We see that posterior draws all go near the observed data points, and in regions where few data points are, there is more variation in what the functions do, since the weight of the basis function in that location is not constrained.

Hyperparameters & marginal likelihoods

The posterior over weights is only a partial solution to the problem, as we are also uncertain about the properties of the prior or likelihood, and we want to infer these properties from data as well. We collect the parameters that the model depends on, for example the number of basis functions M, the scale of the functions σ_w, and the noise σ in the *hyperparameters* θ, and infer these, again using Bayes' rule and the marginal

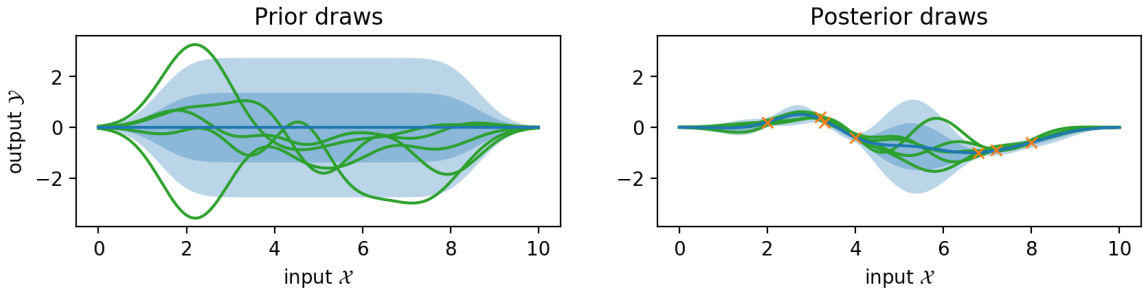


Fig. 1.3 Samples from the prior and posterior for a model with 10 basis functions. The posterior mean is shown in blue, and the blue shaded regions show the marginal 1 and 2 standard deviations of the posterior.

likelihood:

$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}\,. \tag{1.14}$$

While technically this posterior cannot be simplified, and we will need to make predictions by integrating over both θ and $\mathbf{w}$, we will in practice choose a single θ to work with. This is well justified in the case where there are few hyperparameters, since the posterior for θ only will be dominated by the marginal likelihood $p(\mathbf{y}|\theta)$ at its maximum [MacKay, 1999; Rasmussen and Williams, 2005, §5.2]. The posterior $p(\theta|\mathbf{y})$ can then be approximated by a delta function. This greatly simplifies prediction, as it now only requires a single integral over the Gaussian $\mathbf{w}$:

$$p(\theta|\mathbf{y}) \approx \delta(\theta - \hat{\theta}) \qquad \hat{\theta} = \operatorname*{argmax}_{\theta} p(\mathbf{y}|\theta)\,, \tag{1.15}$$

$$\implies \int p(y^*|\mathbf{w})p(\mathbf{w}|\mathbf{y},\theta)p(\theta|\mathbf{y})\mathrm{d}\theta\mathrm{d}\mathbf{w} \approx \int p(y^*|\mathbf{w})p(\mathbf{w}|\mathbf{y},\hat{\theta})\mathrm{d}\mathbf{w}\,. \tag{1.16}$$

1.2.2 Notation

For the following, we will introduce notation that we will use throughout the rest of this thesis. We will be interested in regression and classification problems. Both have an input space of $\mathbb{R}^D$. Outputs for regression outputs are in $\mathbb{R}$, while for classification they are natural numbers indicating the class. We collect the N input vectors $\{\mathbf{x}_n\}_{n=1}^N$ in the matrix $X \in \mathbb{R}^{N\times D}$, and denote the vector of observations at this point as $f(X)$ or $\mathbf{f}$ for short. We also collect the evaluations of all basis functions for an input $\mathbf{x}$ in the vector $\boldsymbol{\phi}(\mathbf{x})$, and for a collection of inputs X in the matrix $\Phi(X)$. We will predict at a point $\mathbf{x}^*$ or points X^*, giving the vector of function evaluations $f(X^*)$ or $\mathbf{f}^*$.

1.2.3 From basis functions to function values

In the previous section we saw that a distribution over weights implied a distribution over functions, and that we can perform inference over the weights to perform inference over the functions. We were interested in the posterior over the weights, since it was sufficient for calculating predictions and the marginal likelihood. Here we show that we can find both the posterior for new points $f(X^*)$, and the marginal likelihood by only considering the joint distribution of function evaluations. This side-steps any manipulations of distributions on the weights.

Prior over function values

In order to obtain the posterior over $f(X^*)$, we first need to obtain their prior, joint with the points required for the likelihood. We start with the deterministic relationship between $f(X)$ and $\mathbf{w}$:

$$\begin{aligned} f(X) &= \begin{bmatrix} f(\mathbf{x}_1) & f(\mathbf{x}_2) & \cdots \end{bmatrix}^\mathsf{T} = \begin{bmatrix} \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_1) & \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_2) & \cdots \end{bmatrix}^\mathsf{T} \\ &= \Phi(X)\mathbf{w}\,. \end{aligned} \tag{1.17}$$

So we can get the joint distribution over any set of function points X by integrating:

$$\begin{aligned} p(f(X)) &= \int p(f(X)|\mathbf{w})p(\mathbf{w})\mathrm{d}\mathbf{w} \\ &= \int \delta(f(X) - \Phi(X)\mathbf{w})p(\mathbf{w})\mathrm{d}\mathbf{w} \\ &= \mathcal{N}(f(X); \boldsymbol{\mu}, K_{\mathbf{ff}})\,. \end{aligned} \tag{1.18}$$

Where the mean and covariance of the distribution are given by:

$$\boldsymbol{\mu} = \Phi(X)\mathbb{E}_{\mathbf{w}}[\mathbf{w}] = 0 \tag{1.19}$$

$$\begin{aligned} K_{\mathbf{ff}} &= \mathrm{Cov}_{\mathbf{w}}[\Phi(X)\mathbf{w}] = \Phi(X)\mathbb{E}_{\mathbf{w}}[ww^\mathsf{T}]\Phi(X)^\mathsf{T} \\ &= \sigma_w^2\Phi(X)\Phi(X)^\mathsf{T}\,. \end{aligned} \tag{1.20}$$

$\Phi(X)\Phi(X)^\mathsf{T}$ is the matrix containing all pairs of inner products of basis-function evaluation vectors $\boldsymbol{\phi}(\mathbf{x})$. We can alternatively view each element of $K_{\mathbf{ff}}$ as being evaluated by a function giving the inner product between the feature vectors of two points. This function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is named the *kernel* or *covariance function*

(more details later), and in this case is given by:

$$k(\mathbf{x}_i, \mathbf{x}_j) = [\Phi(X)\Phi(X)^\mathsf{T}]_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_j) = \sum_{b=1}^{B} \phi_b(\mathbf{x}_i)\phi_b(\mathbf{x}_j)\,. \tag{1.21}$$

In general we denote the covariance of the distribution of the N function values $f(X)$ (or $\mathbf{f}$ for short) with $K_{\mathbf{ff}} \in \mathbb{R}^{N\times N}$. Through the same argument as above, we can also get the joint covariance between two separate vectors of function values, e.g. $f(X)$ and $f(X^*)$, denoted by $K_{(\mathbf{f},\mathbf{f}^*)(\mathbf{f},\mathbf{f}^*)}$. The covariance between $f(X)$ and $f(X^*)$ will be given denoted by $K_{\mathbf{ff}^*}$ and is the $N \times N^*$ sub-matrix of $K_{(\mathbf{f},\mathbf{f}^*)(\mathbf{f},\mathbf{f}^*)}$.

Posteriors and predictions

We now have our *equivalent* prior to the one defined in weight space. If we consider the joint prior for function values of the inputs we observe, plus those we want to predict at, we can get the posterior and predictive distribution in one go:

$$\begin{aligned} p(f(X), f(X^*)|\mathbf{y}) &= \frac{p(\mathbf{y}|f(X))p(f(X), f(X^*))}{p(\mathbf{y})} \\ &= \frac{p(\mathbf{y}|f(X))p(f(X))}{p(\mathbf{y})} p(f(X^*)|f(X)) \qquad (1.22) \\ &= p(f(X)|\mathbf{y})p(f(X^*)|f(X))\,. \qquad (1.23) \end{aligned}$$

Interestingly, we see that by applying the product rule[8] to the joint prior (left hand side) we can split the joint posterior into the posterior for the observed function values, and the points we want to predict at. This means that we can get the posterior over any set of new inputs simply from the posterior over $f(X)$![9] In a sense, this gives us access to properties of the distribution over the entire latent function, just as the posterior over $\mathbf{w}$ did. This isn't odd, since the way we constructed the prior $p(f(X), f(X^*))$ meant that it had a consistent function underlying it.

[8]The exact form of the conditional Gaussian $p(f(X^*)|f(X))$ can be found in Rasmussen and Williams [2005, p.16].

[9]Kernel Ridge Regression (a similar non-Bayesian technique), has the similar property that the "optimal" solution can be represented with basis functions centred only at the observed points. This is known as the *Representer Theorem* [Kimeldorf and Wahba, 1970]. Csató and Opper [2002] also note this link.

1.2.4 Gaussian processes

So far, we have specified priors over functions indirectly by specifying a prior distribution over weights in a basis function model. We saw that either the posterior over the weights or over the observed function values was sufficient for making predictions over any point on the rest of the latent function. One advantage of the function-value view is computational, if the number of basis functions is large enough. Manipulating a Gaussian posterior incurs a cost that scales with the cube of the number of variables considered. So if there are fewer data points than basis functions, there will be a computational speed-up.

From marginal distributions to functions

We argued that because we could define the prior over an arbitrarily large set of function values, we had a handle on the distribution of the overall function. This argument is mathematically formalised by the *Kolmogorov extension theorem*. Informally[10], the theorem states that if one can define joint distributions on finite subsets of some input space $\mathcal{X}$ which are all *marginally consistent*, then there exists a probability measure of the whole space. By marginally consistent, we mean that we obtain the same distribution if we directly define a joint over some set of points, as if we first define a distribution over a larger set, and then marginalise out the appropriate variables:

$$p_{marginal}(f(X)) = \int p(f(X), f(X^*))\mathrm{d}f(X^*) = p_{direct}(f(X))\,. \tag{1.24}$$

The beauty of this is that it shows that our basis function model does in fact define a distribution over entire functions. Marginal consistency simply comes from the property of the Gaussian distribution that marginalisation of a variable entails dropping the corresponding row and column from the mean and covariance.

Additionally, this view leads to a different method of specifying distributions over functions. In the previous section, the covariance function $k(\mathbf{x}, \mathbf{x}')$, and matrix $K_{\mathbf{ff}}$ were constructed from a basis function model. The Kolmogorov extension theorem above implies that this construction is redundant. It is equally valid to specify some function $k(\mathbf{x}, \mathbf{x}')$ directly, and it will specify a distribution over functions, provided the distributions it implies over finite dimensional marginals are consistent. We call this "distribution over functions" with Gaussian marginals a *Gaussian process*. It is completely defined by its covariance function $k(\mathbf{x}, \mathbf{x}')$ and mean function $\mu(\mathbf{x})$, which

[10]For a formal discussion focused on GPs, see Matthews [2016], or Seeger [2004] for something in between.

determines the means of the marginal Gaussian distributions[11]. We will denote a function distributed according to a Gaussian process as

$$f \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))\,. \tag{1.25}$$

And, as discussed, the distribution over N function values $f(X)$ at inputs X will be Gaussian distributed:

$$f(X) \sim \mathcal{N}\left(f(X); \begin{bmatrix}\mu(\mathbf{x}_1)\\ \mu(\mathbf{x}_2)\\ \dots \\ \mu(\mathbf{x}_N)\end{bmatrix}, \begin{bmatrix}k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \dots & k(\mathbf{x}_1,\mathbf{x}_N)\\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \dots & k(\mathbf{x}_2,\mathbf{x}_N)\\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N,\mathbf{x}_1) & k(\mathbf{x}_N,\mathbf{x}_2) & \dots & k(\mathbf{x}_N,\mathbf{x}_N)\end{bmatrix}\right). \tag{1.26}$$

Above, we connected the prior over function values to a Gaussian process over functions. If we assume the regression case with the Gaussian likelihood

$$p(\mathbf{y}|f(X)) = \prod_{n=1}^{N} p(y_i|f(\mathbf{x}_i)) = \prod_{n=1}^{N} \mathcal{N}\left(y_i; f(\mathbf{x}_i), \sigma^2\right), \tag{1.27}$$

we get a Gaussian posterior $p(f(X), f(X^*)|\mathbf{y})$ from equation 1.23.[12] We can work backwards from this to find that the posterior implies a Gaussian process as well:

$$f(\mathbf{x}^*)|\mathbf{y} \sim \mathcal{GP}\left(k_{\mathbf{f}^*\mathbf{f}}\left(K_{\mathbf{ff}} + \sigma^2 I\right)^{-1}\mathbf{y}, K_{\mathbf{f}^*\mathbf{f}^*} - k_{\mathbf{f}^*\mathbf{f}}\left(K_{\mathbf{ff}} + \sigma^2 I\right)^{-1} k_{\mathbf{ff}^*}\right). \tag{1.28}$$

Kernels & basis functions

From this *function space view*, we specify the properties of the latent function through the covariance/kernel function, instead of the number and shape of the basis functions and prior over the weights. Properties like differentiability, expected change over a distance and periodicity can all be determined – see figure 1.4 for some samples from different kernels.

Kernels can not just be constructed from explicit feature spaces, but feature spaces can also be specified by a kernel. Mercer's theorem [Rasmussen and Williams, 2005,

[11] The mean function of the Gaussian process constructed by the basis function model was constantly zero. All derivations follow through with non-zero means, as the marginal consistency property is unchanged.

[12] See Rasmussen and Williams [2005, p.16] for the full expressions.

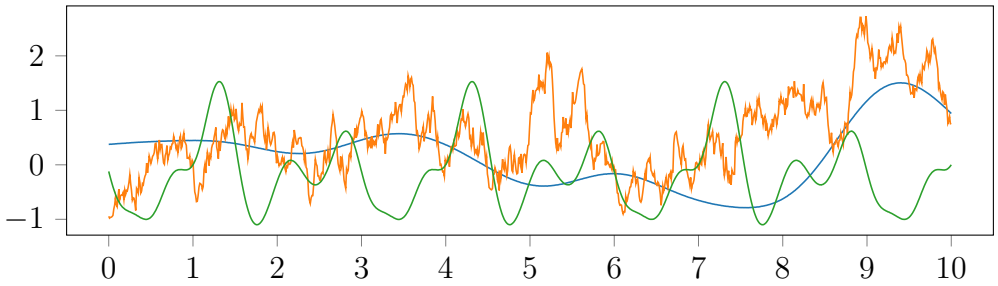


Fig. 1.4 Samples from a squared exponential (blue), Matérn-1/2 (orange), and periodic squared exponential kernel (green).

§4.3] states that any kernel can be written as an infinite series summation:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i^*(\mathbf{x}) \,. \tag{1.29}$$

For many kernels, $\lambda_i = 0$ for i being greater than some B, showing that a kernel can be written as an inner product between vectors, as is the kernel constructed from a feature space explicitly in equation 1.21. The number of non-zero λ_is determines how many basis functions the Gaussian process effectively has. Interestingly, certain kernels have $B = \infty$, implying an *infinite* number of basis functions. These kernels are said to be *non-degenerate* and imply a *non-parametric* model. In fact, if we take the limit of an infinite number of basis functions in section 1.2.1, we obtain a Gaussian process prior with the "squared exponential" (SE) kernel [Rasmussen and Williams, 2005, §4.2.1]:

$$k_{sqexp}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2l_d^2} \right) . \tag{1.30}$$

1.2.5 How many basis functions?

So far, we saw that Gaussian processes can be specified by their basis functions and weight prior, or directly using a kernel, and that kernels can imply an infinite number of basis functions. But how many basis functions do we need, and what is the advantage of having an infinite number? The number of basis functions influences both:

- the range of functions that the Gaussian process can learn,
- the predictive uncertainties of the posterior.

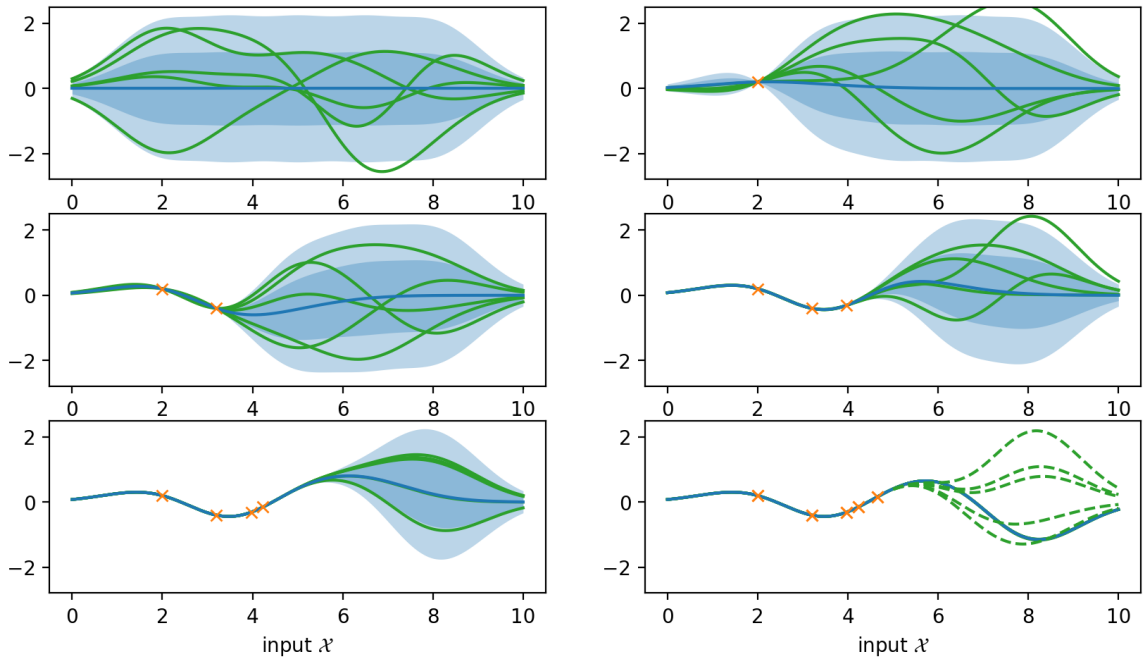


Fig. 1.5 Prior conditionals for a 5 basis function model. The shaded regions indicate the variance of $p\left(f(\mathbf{x}_n)|\{f(\mathbf{x}_i)\}_{i=1}^{n-1}\right)$.

Limited capacity of finite basis function models

Models with a finite number of basis functions can only produce a finite number of linearly independent functions. This is a very restricted set, so it is easy to construct functions which lie outside this prior distribution. We will illustrate this in both the weight and function value view. We start by noting that the probability of observing the function values $f(X)$ under the prior $p(f(X))$ can be expanded as

$$\begin{aligned} p(f(X)) &= p(f(\mathbf{x}_1))p(f(\mathbf{x}_2)|f(\mathbf{x}_1))p(f(\mathbf{x}_3)|f(\mathbf{x}_2), f(\mathbf{x}_1))\dots \\ &= \prod_n p\left(f(\mathbf{x}_n)|\{f(\mathbf{x}_i)\}_{i=1}^{n-1}\right). \end{aligned} \tag{1.31}$$

This shows that we can analyse the prior and calculate the probability density of a set of function values by the one dimensional prior conditionals. In figure 1.5 we visualise the prior conditionals for an increasing set of points we condition on. We see that after conditioning on 5 points, the variance of $p\left(f(\mathbf{x}_6)|\{f(\mathbf{x}_i)\}_{i=1}^{5}\right)$ becomes zero, indicating that any function with any function value that is not predicted has zero probability under the prior.

We can analyse this behaviour in more generality by considering either the posterior over the weights $p(\mathbf{w}|\mathbf{f})$ or the prior $p(\mathbf{f})$ directly. We can find the posterior for

the weights by writing the deterministic relationship between the weights $\mathbf{w}$ and the function values $\mathbf{f}$ from equation 1.17 as a delta function likelihood:

$$p(\mathbf{w}|\mathbf{f}) = \frac{p(\mathbf{f}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{f})} = \frac{\delta(\mathbf{f} - \Phi(X)\mathbf{w})\mathcal{N}(\mathbf{w}; 0, \sigma_w^2)}{p(\mathbf{f})}\,. \tag{1.32}$$

The resulting posterior is the prior density projected onto the plane $\mathbf{f} - \Phi(X)\mathbf{w} = 0$:

$$p(\mathbf{w}|\mathbf{f}) = \mathcal{N}\Big(\mathbf{w}; \Phi(X)^\intercal(\Phi(X)\Phi(X)^\intercal)^{-1}\mathbf{f}, \sigma_w^2\Big(I - \Phi(X)^\intercal(\Phi(X)\Phi(X)^\intercal)^{-1}\Phi(X)\Big)\Big)\,. \tag{1.33}$$

Each new observation gives another linear constraint on $\mathbf{w}$, until we have M observations, when the variance of $p(\mathbf{w}|\mathbf{f})$ becomes zero. At this point, there are no more degrees of freedom in the prior and function values are either consistent with the others, or the prior does not contain a function which can pass through the required points.

This same insight can be gained from the prior $p(\mathbf{f})$ directly. The covariance is given by $K_{\mathbf{ff}} = \sigma_w^2\Phi(X)\Phi(X)^\intercal$ from equation 1.20. This covariance matrix has a maximum rank of M since $\Phi(X) \in \mathbb{R}^{N\times M}$. As a consequence, if we consider more points than basis functions ($N > M$), it will necessarily have a zero eigenvalue. Any vector of function values $\mathbf{f}$ with a component in the corresponding eigenvector, will have zero density under the prior.

Using infinite basis functions

In the previous section we saw that a finite basis function model only has enough degrees of freedom for the function to be fully specified at a finite number of points. This leads to the question of how many basis functions to use in a model. A model with a small number of bases can be expected to fit model a small dataset well, with more basis functions being needed as more data arrives, and more complex structure is observed. If the number of bases is treated as a hyperparameter, the marginal likelihood could be used for model selection. Rasmussen and Ghahramani [2001] show that this procedure is unsatisfactory, as the marginal likelihood does not distinguish between models once enough basis functions are added. Furthermore, they show that in this regime, it is the shape of the basis functions and the prior over their weights that *is* distinguished by the marginal likelihood. Because of this, they argue that the convenient and correct way to choose the number of basis functions, is to take an infinite number of them. This also makes sense when considering the case of modelling a single phenomenon, but with an increasingly large dataset. We expect a single model

to explain the phenomenon, since the size of the dataset does not change the process we are trying to model. Using the marginal likelihood to determine the size of the model makes our inference depend on a detail that is irrelevant to the phenomenon we are actually modelling. This also leads to a contradiction with the Bayesian framework, where the prior distribution should be consistent with our beliefs, and determined before observing any data, as we a-priori believe each finite basis function model will be falsified after observing enough data, since none of them will contain the exact function of interest, which would require us to place a 0 prior probability on them.

Often, the consequence of having a prior which has such a broad coverage of functions, is that the method will be *universally consistent* [Rasmussen and Williams, 2005, §7.1,§7.2], meaning that any function can be approximated to arbitrary precision. Not all kernels with an infinite number of basis functions exhibit this property (we see an example of that in chapter 5), but many common ones like the squared exponential (equation 1.30) do. These kernels are named *universal* [Micchelli et al., 2006]. Working with consistent algorithms can be very desirable, since we are guaranteed to improve to an optimal result as we observe more data.

Influence on uncertainty

It may also be dangerous to use a limited number of basis functions from the point of view of getting good predictive uncertainties. In figure 1.3 we see that the predictive uncertainties decrease outside the range $[2 \dots 8]$, in both the prior and posterior. Without a basis function in a location, the model is blind to the fact that it may be uncertain to what is going on there. This is also illustrated by the zero eigenvalues in $K_{\mathbf{ff}}$ which arise from parametric models. The model not only cannot learn functions that lie in that subspace, but it can also not represent uncertainty in these directions.

By taking the infinite limit, resulting in the squared exponential kernel, we get the more sensible error bars seen in figure 1.6. Sensible error bars in data-sparse regions can be very important not just for avoiding over-confident predictions, but also for assessing how much there is to learn. Particularly reinforcement learning algorithms [Deisenroth and Rasmussen, 2011] have a need for accurate uncertainties, as this indicates how much more could be to learned, which helps to create robust control policies and for guiding exploration.

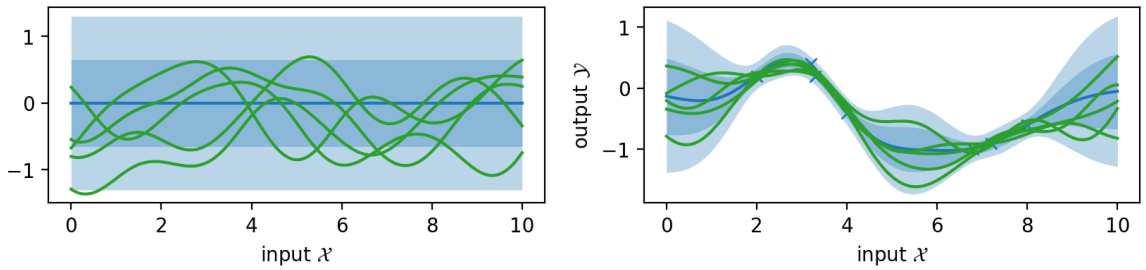


Fig. 1.6 Samples from the prior and posterior over functions for a squared exponential kernel with infinite basis functions. The posterior mean is shown in blue, and the blue shaded regions show the marginal 1 and 2 standard deviations of the posterior, and samples in green.

1.3 Why approximate non-parametric models?

The previous sections described the advantages of the Bayesian framework and using non-parametric models. The Bayesian framework is desirable because it provides a unified way of:

- **Learning under uncertainty caused by lack of data** by using posterior distributions to represent uncertainty.
- **Selecting or inferring assumptions or properties of a model** through hyperparameter optimisation of the marginal likelihood, or joint Bayesian inference.

Non-parametric models fit particularly well within this framework, both practically and philosophically, as they provide:

- **A sound way of modelling phenomena, irrespective of dataset sizes.** When considering an increasingly large training dataset, non-parametric models avoid the problem of needing to change the model prior by adding modelling capacity (in this case, basis functions) to deal with the increasing complexity that has been revealed.
- **A way to ensure correctly large uncertainty estimates in regions with little data.** The limited capacity of parametric models causes weight uncertainty to be insufficient for representing uncertainty where there is no data, as there is no incentive to place basis functions that are unconstrained by the data in those regions. And if there is no basis function, there can be no variation in the function.

This thesis is concerned with developing practical methods which do not sacrifice the benefits from either the Bayesian or the non-parametric modelling approaches. In order to obtain practical methods we follow earlier work which approximates the posteriors of non-parametric Gaussian processes with a limited number of basis functions. Now, it may seem strange to go through a lot of effort to set up a model with an infinite number of basis functions, only to approximate it with a limited number again. Is this not equivalent to using a finite model in the first place?

As we will see in followings sections, there is a difference between a finite approximation to a non-parametric model, and a parametric model. It is possible to maintain the desirable properties discussed above in approximations which have much lower computational cost, while either maintaining the desired benefits, or at least having an indication that the approximation has failed.

1.4 Main contributions

In this thesis, we are interested in assessing and improving the state of approximations in non-parametric Gaussian process models. We summarise the contributions of each chapter:

Chapter 2 We review the main sparse approximations in Gaussian processes, and propose a list of desiderata for approximations to non-parametric models.

Chapter 3 We evaluate two popular Gaussian process approximations – FITC and VFE – and examine their behaviour in detail. We show that FITC, while being a usable machine learning method, has some pathologies which raise questions about its suitability for approximating GP models. We additionally discuss the consequences of the VFE objective also being a lower bound to the FITC marginal likelihood.

Chapter 4 We extend inter-domain Gaussian process approximations to give them more basis functions for the mean, at a lower cost compared to traditional inducing point approximations. We show that, while the extra capacity can improve performance, the lack of flexibility in the approximate covariance hinders hyperparameter selection, which limits the usability of the approximation in some cases. We do show practical use for compressing GP models, by showing improved predictive performance for limited storage space.

Chapter 5 We introduce an inter-domain approximation for effectively performing inference in Gaussian process models with invariances encoded into their priors. We use this technique to build a non-parametric convolutional Gaussian process priors and show their use in improving the classification of images. We also discuss the relation of convolutional Gaussian processes to convolutional neural networks, particularly as an infinite limit.

Chapter 6 We address some pedagogical issues regarding random-input GPs and Gaussian process State Space Models (GPSSMs) in particular.

Chapter 7 We review our contributions, and discuss them in relation to the wider field of machine learning.

Chapter 2

Sparse Gaussian process approximations

Although Gaussian processes have many desirable properties from a modelling point of view, they become computationally intractable to manipulate for even moderately sized datasets. Interestingly, the source of the intractability in GP models is orthogonal to the usual reasons posteriors may be intractable (mentioned in section 1.1.4). This is best illustrated by Gaussian process regression [Rasmussen and Williams, 2005, §2.2], and we will take this as a running example. Although the distributions of interest (i.e. the marginal likelihood, posterior, predictive and even the prior), are all Gaussian and *analytically tractable*, they become *computationally* intractable because they and their derivatives contain matrix operations[1] costing $\mathcal{O}(N^3)$, where N is the number of observations on the GP. In order to circumvent this prohibitive computational constraint, low-rank approximations have been developed which aim to approximate the full Gaussian process without having to perform $\mathcal{O}(N^3)$ operations. In most cases, this involves approximating $K_{\mathbf{ff}}$ by a rank $M < N$ matrix which can be manipulated in $\mathcal{O}(NM^2)$. Generally, these methods can be seen as learning about a small number (M) quantities that are highly informative of what the posterior Gaussian process is doing more globally. Ferrari-Trecate et al. [1999] and Williams et al. [2002] gave evidence showing that GPs could be well approximated this way (also discussed in Rasmussen and Williams [2005, §7.1]). These quantities can be the weights of explicit basis functions [Lázaro-Gredilla et al., 2010; Rahimi and Recht, 2008], the function value at certain input locations [Quiñonero-Candela and Rasmussen, 2005; Titsias,

[1] Specifically, a log determinant and matrix inversion for $K_{\mathbf{ff}}$, or alternatively, a single Cholesky decomposition of $K_{\mathbf{ff}}$.

2009b], or more abstract representations [Lázaro-Gredilla and Figueiras-Vidal, 2009] related to the Gaussian process.

In the following sections we will discuss several methods that have been proposed to approximate Gaussian processes. We will refer to the target of our approximation as the "exact" Gaussian process, which will be a Gaussian process with some given non-degenerate kernel function $k(\mathbf{x}, \mathbf{x}')$. We will discuss each of the approximations in relation to some desiderata.

2.1 Desiderata for non-parametric approximations

In the previous chapter, we laid out the advantages of the Bayesian framework for machine learning, and non-parametric models. However, given realistic computational constraints, we need to make approximations in order to get usable answers. In this thesis, we are interested in approximations which allow us to retain the advantages of the Bayesian approach, and non-parametric Gaussian processes. To retain advantages of Bayesian methods, we mainly want to be able to separate the effects of the prior modelling assumptions and the approximation, so we can use marginal likelihoods for model comparison and as a guide to build new models.

For Gaussian process models, we believe that these advantages are maintained if an approximation satisfies the following desiderata. We desire that approximations to non-parametric Gaussian processes:

1. Allow for some form of assessment of the distance to the exact solution (of both the posterior and marginal likelihood), within the same computational budget as finding the approximation.

2. Recover the exact solution increasingly well given more computational resources, and ideally exactly in some limit.

3. Obtain error bars that behave as the non-parametric model.

We accept that these requirements may preclude some methods which can be very practically useful in certain situations. However, we believe these requirements to be essential for obtaining the advantages outlined earlier. We will discuss each desideratum in turn to explain its necessity.

Assessment of the approximation As mentioned earlier, GP approximation methods use M quantities to summarise the approximation. This number allows the computational requirements to be traded off against accuracy. This value needs to be chosen,

and it should be set to the smallest value possible. To set this free parameter in a principled way, we need to quantify how good the approximation is for a given M, and whether it would be useful to increase it at all. Therefore, we would like some way to quantify the distance to the exact solution. For this quantity to be practically useful, it also needs to be computable at roughly the same cost as the approximation itself. A measure relying on computation of the intractable exact solution, for example, is not useful.

Recovery of the exact solution As more resources are added, we want the behaviour of the approximation to grow closer to the exact solution, ideally monotonically. Firstly, this guarantees that adding computational resources gives a better answer, even when desideratum 1 is not satisfied. Secondly, this indicates that an improvement in performance that is observed after improving the approximation can be attributed to qualities that the exact solution possesses.

If the approximation diverges from the exact solution, it is not suitable for use within the Bayesian framework, even if it achieves good performance. A central part of Bayesian analysis, is to perform model comparison to check whether prior assumptions made in the model are compatible with the observations. If the approximation does not accurately recover the true posterior and marginal likelihoods, any good or bad performance may be more attributable to deviations introduced by the approximation, rather than assumptions made in the prior. Later in this thesis we will see an approximation which perform well in certain cases, but not because it accurately resembles the desired solution.

Non-parametric uncertainties One large advantage of some fully non-parametric Gaussian processes, is that their error bars grow far away from the data (see section 1.2.5 & figure 1.6), avoiding overconfidence if the model is faced with a prediction task from e.g. a different input distribution. We would like to keep this property in addition to good approximations of the marginal likelihood and posterior on a test set.

In the following sections, we will review some existing approximations and assess them based on the above desiderata.

2.2 Explicit feature representations

2.2.1 Random feature expansions

While earlier (section 1.2.3) we saw the advantages of working with kernels rather than with the basis functions directly, Rahimi and Recht [2008] suggest that randomly chosen features with a carefully chosen distribution can give rise to a kernel that is very close to some desired kernel. The idea comes from Bochner's theorem (see Rasmussen and Williams [2005, §4.2.1]), which (informally) states that any stationary kernel has a positive and normalisable Fourier transform. With appropriate rescaling, the Fourier transform can be interpreted as a density, which can then be sampled from. The resulting features will be sines and cosines with random frequencies. Rahimi and Recht [2008] prove the following statement (reproduced, but adapted to the notation used here):

Theorem (Uniform convergence of Random Fourier features)**.** *Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter* $\mathrm{diam}(\mathcal{M})$*. Then for the mapping to the feature space $\phi(\cdot)$ obtained from the random features, the following holds:*

$$\mathcal{P}\left(\sup_{x,y\in\mathcal{M}} |\phi(\mathbf{x})^\intercal\phi(\mathbf{x}') - k(\mathbf{x},\mathbf{x}')| \geq \epsilon\right) \leq 2^8\left(\frac{\sigma_p \mathrm{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{4(d+2)}\right) \quad (2.1)$$

Where $\mathrm{diam}(\mathcal{M})$ *is the supremum over distances between pairs of points in $\mathcal{M}$, σ_p is the variance of the distribution where the random frequencies are sampled from, D is the number of feature vectors and d is the dimensionality of the input space.*

This is a great result, showing that it is highly likely that within some limited region $\mathcal{M}$, evaluations of the approximate kernel will lie close to their true value. Le et al. [2013] provide algorithmic improvements and show that good performance can be obtained with few basis functions. In some cases the approximation even outperformed the exact method.

2.2.2 Optimised features

Lázaro-Gredilla et al. [2010] present a different take on this idea, pointing out that the random features also can just be seen as defining a new kernel, which is parameterised by the chosen frequencies. While Rahimi and Recht [2008] take the view that the approximation is exact in the limit of infinite samples, Lázaro-Gredilla et al. [2010] choose to view these frequencies as kernel hyperparameters which can be optimised

over to fit the data better. This "Sparse Spectral Gaussian process" squeezes out extra performance per basis function, at the expense of optimisation during training.

2.2.3 Desiderata

Both methods essentially work by introducing a new parametric model that makes some concessions to key properties of the original model. Both methods have lost their non-parametric error bars. Because of the sinusoidal basis functions, all functions in the posterior will be periodic, resulting in locations far from the data where the variance collapses to what it is near the data and the function repeats itself. If enough basis functions are used, the periods can be made large enough such that this won't pose a practical problem. However, this may be hard to assess, particularly in high dimensions. Regardless of practical ways to mitigate this issue, this is a qualitative difference the approximation introduces, which would be nice to avoid.

The random feature expansion monotonically improves given more features, and is guaranteed to converge to the true model in the limit of infinite features. The bound on the deviation of the kernel is easily computable, but a bit limited since the posterior depends on $K_{\mathbf{ff}}$ *inverse*, while the bound only guarantees things about elements of $K_{\mathbf{ff}}$. Additionally, the guarantee only holds for points within $\mathrm{diam}(\mathcal{M})$.

Optimising the features removes these guarantees. While the advantage is a more compressed representation (more performance for less basis functions), Lázaro-Gredilla et al.'s [2010] approximation does not get closer to the original kernel with more basis functions. The method is better seen as a new model in itself.

We emphasise though, that pointing out these "flaws" does not constitute an argument against the impressive practical utility of these algorithms. However, these properties are important within the question of this thesis of how well posteriors of non-parametric Gaussian processes can be approximated, and which properties get preserved.

2.3 Inducing point approximations

In inducing point approximations, the M quantities that are learned are function values (*inducing outputs*) at specific input locations (*inducing inputs*), rather than the weights of some basis functions. In section 1.2.3 we saw that, strictly speaking, the posterior for all observed function values is needed to make predictions, and that a small set of M quantities is not sufficient. The main intuition that underpins this approximation,

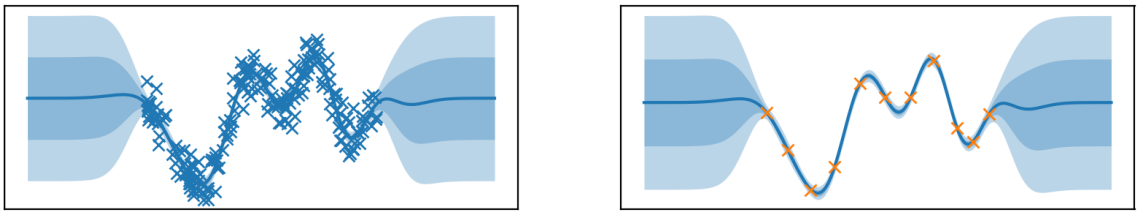


Fig. 2.1 An example of a posterior obtained from many noisy observations (left) and from very few noiseless observations (right), on Snelson and Ghahramani's [2006] commonly used dataset. The two posteriors are visualised with their means and 1 and 2 σ credible intervals, and are almost identical.

is that there is a lot of redundant information in representing the distribution of many function values with noisy observations in the same region. Because the GP prior places such strong constraints on what values neighbouring outputs can take, such a distribution would be represented almost as well by learning very well about a single point and then using the prior to constrain the distribution on neighbouring points (figure 2.1). The goal for the approximation method then becomes: "*what M inputs and strongly constrained outputs would result in a posterior close to the true GP posterior?*"

Over the next few sections, we will review different views on inducing point approximations, and some of the methods that have been proposed.

2.3.1 As likelihood approximations

Inducing point approximations can be created using a similar approach as in the previous section: by introducing an approximate model with the desired computational properties, which in some way has properties similar to the exact model. Seeger et al. [2003] and Snelson and Ghahramani [2006] presented inducing point approximations based on approximations to the likelihood.

They started by concentrating on obtaining the posterior over M inducing points, based on the observation that this would at least speed up predictions. Even within the framework of exact Gaussian processes, this is a completely valid question to ask the model. We are simply querying the posterior over the full latent function at some marginals, determined by the inputs Z:

$$p(f(Z)|\mathbf{y}, X) = \frac{p(\mathbf{y}|f(Z), X)p(f(Z))}{p(\mathbf{y}|X)}\,. \tag{2.2}$$

Following on the notation from the previous chapter, we refer to $f(Z)$ as $\mathbf{u}$ for short.

The likelihood $p(\mathbf{y}|f(Z))$ has to be obtained by marginalising over $f(X)$. For regression, this is analytically tractable and gives the following likelihood and posterior:

$$p(\mathbf{y}|\mathbf{u}, X) = \int \prod_i p(y_i|f_i) p(\mathbf{f}|\mathbf{u}, X, Z) \mathrm{d}\mathbf{f} \tag{2.3}$$

$$= \mathcal{N}\Big(\mathbf{y}; K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} \mathbf{u}, K_{\mathbf{ff}} - K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uf}} + \sigma^2 I\Big)\,, \tag{2.4}$$

$$p(\mathbf{u}|\mathbf{y}, X) = \mathcal{N}\Big(\mathbf{u}; K_{\mathbf{uf}}(K_{\mathbf{ff}} + \sigma^2)^{-1}\mathbf{y}, K_{\mathbf{uu}} - K_{\mathbf{uf}}\Big(K_{\mathbf{ff}} + \sigma^2 I\Big)^{-1} K_{\mathbf{fu}}\Big)\,. \tag{2.5}$$

Finding this smaller posterior still does not remove the large burden of dealing with an $N \times N$ matrix. The first approximation is introduced in the likelihood of $\mathbf{u}$, as a factorisation constraint over each observation. The resulting "approximate" likelihood is denoted by $q(\mathbf{y}|\mathbf{u})$:

$$q(\mathbf{y}|\mathbf{u}, X) = \mathcal{N}\Big(\mathbf{y}; K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} \mathbf{u}, \mathrm{diag}[\Sigma] + \sigma^2 I\Big)\,. \tag{2.6}$$

With application of the Woodbury matrix identity, an inversion of a full $N \times N$ matrix is avoided, resulting in the desired computational saving. Different approximations will have different values for Σ, which will be discussed later.

2.3.2 As model approximations

Quiñonero-Candela and Rasmussen [2005] present an alternative, unifying view on the "likelihood approximation" view. Instead, they show that the approximations can be seen as performing *exact* Bayesian inference, only with a modified prior and the original likelihood. In addition to providing a unified framework to compare these methods, this interpretation is also more natural since it allows the modified prior to be analysed while retaining the well-developed intuition about the i.i.d. Gaussian likelihood.

The unifying view shows that all methods can be constructed by a generative process that samples the inducing outputs $\mathbf{u}$ first, followed by independent draws for the training $\mathbf{f}$ and testing values $\mathbf{f}^*$ (figure 2.2). In all approximations, $\mathbf{u}$ is drawn from the exact prior. The approximations differ, and are characterised by their *training conditional* $q(\mathbf{f}|\mathbf{u})$, and *testing conditional* $q(\mathbf{f}^*|\mathbf{u})$, which can be derived from the approximate likelihood view and the proposed prediction procedure.

The most powerful insight gained from this unifying view, is that the GP approximations can themselves be seen as models in their own right, where exact inference is performed. The Gaussian conditionals imply a Gaussian joint prior $p(\mathbf{f}, \mathbf{f}^*)$, where the

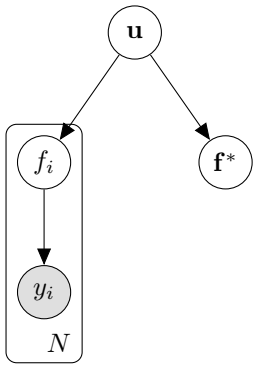


Fig. 2.2 Independence assumptions for approximate sparse GP models.

conditional independence property implies that inference over the M inducing variables $\mathbf{u}$ is sufficient for predicting the points $\mathbf{f}^*$.

Two approximations discussed in Quiñonero-Candela and Rasmussen [2005] are of particular interest in the context of this thesis: the Deterministic Training Conditional (DTC), and the Fully Independent Training Conditional (FITC). We will summarise these methods, following the unifying framework, while discussing their properties in relation to our desiderata from non-parametric approximations.

2.3.3 Consistency with an approximate GP

Although the sparse GP approximations can definitely be seen as performing exact inference with some finite-dimensional Gaussian, one can ask the question whether the procedure is also performing inference in a Gaussian process model. For this to be the case, the implied joint prior $p(\mathbf{f}, \mathbf{f}^*)$ has to be marginally consistent in the sense that was laid out in section 1.2.4. In sparse models with the structure in figure 2.2, this boils down to requiring identical training and inducing conditionals. To show this, we can simply consider the marginal distribution of the function at some inputs X.[2] If this depends on whether they are considered a training or testing points, i.e. whether they are in $\mathbf{f}$ or $\mathbf{f}^*$, then there is no consistent distribution over latent functions linking $\mathbf{f}$ and $\mathbf{f}^*$.

$$\int q_{training}(f(X)|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \stackrel{?}{=} \int q_{testing}(f(X)|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \tag{2.7}$$

The marginals will only be consistent if $q_{training}$ equals $q_{testing}$.

[2]Here we depart from the notation of $\mathbf{f} = f(X)$ and $\mathbf{f}^* = f(X^*)$ as the function outputs at training and testing inputs, in order to explicitly highlight that we're considering some arbitrary input $\mathbf{x}$ as either a testing or training point.

It is not necessarily a practical problem if the marginals are not consistent and the method cannot be seen as performing exact inference in a modified Gaussian process. Regardless of what the method is to obtain a posterior distribution over $\mathbf{u}$, a valid Gaussian process over predictions will be induced by the testing conditional. However one can wonder why a predictive model would work well with the posterior of a different model. This definitely is not true in general (imagine applying a posterior over weights to a model with different basis functions), however in the following approximations the difference between training and testing conditions are not too severe.

2.3.4 Deterministic Training Conditional (DTC) approximation

The Deterministic Training Conditional approximation was originally introduced by Seeger et al. [2003]. The method was introduced using a deterministic link between the inducing outputs $\mathbf{u}$ and the points required by the likelihood $\mathbf{f}$, hence its name. The training conditional can be written as

$$q(\mathbf{f}|\mathbf{u}, X) = \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, 0\Big)\,. \tag{2.8}$$

Due to this deterministic nature, this method behaves as a parametric basis function model during training, discussed as Subset of Regressors (SoR) in Quiñonero-Candela and Rasmussen [2005]. $K_{\mathbf{fu}}$ can be interpreted as the evaluation of the M basis functions at the training input locations, with $K_{\mathbf{uu}}^{-1}\mathbf{u} \sim \mathcal{N}(0, K_{\mathbf{uu}}^{-1})$ being the weights. This expression is obtained from $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}; 0, K_{\mathbf{uu}})$.

After obtaining the posterior over $\mathbf{u}$, predictions are made using the testing conditional, which is chosen to be the exact GP's conditional. If we denote the posterior distribution of $\mathbf{u}$ to be $\mathcal{N}(\mu, \Sigma)$, we can also express the full predictive distribution:

$$q(\mathbf{f}^*|\mathbf{u}, X) = p(\mathbf{f}^*|\mathbf{u}, X) = \mathcal{N}\Big(\mathbf{f}^*; K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\mathbf{f}^*\mathbf{f}^*} - K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}^*}\Big)\,, \tag{2.9}$$

$$q(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}\Big(\mathbf{f}^*; K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}, K_{\mathbf{f}^*\mathbf{f}^*} - K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}} - \Sigma)K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}^*}\Big)\,. \tag{2.10}$$

Quiñonero-Candela and Rasmussen [2005] state that using the exact testing conditional "*reverses the behaviour of the predictive uncertainties, and turns them into sensible ones*", compared to SoR, by which was meant that the predictive error bars grow far away from the data (figure 2.3). We would like to unpack this statement a bit further, and contrast the DTC predictive distribution with that of a fully non-parametric GP.

It is clear that the mean will only ever have M degrees of freedom, with its weights determined by $\boldsymbol{\mu}$. In this sense, the predictive distribution does not have the non-parametric property of being a universal approximator. However, the behaviour of the predictive covariance is closer to that of the fully non-parametric GP. Apart from the marginal variances growing as pointed out by Quiñonero-Candela and Rasmussen [2005], samples drawn from the predictive distribution will have infinite degrees of freedom (figure 2.3). This corresponds much more closely with a non-parametric model, in the sense that the posterior correctly shows that there is still structure to be learned.

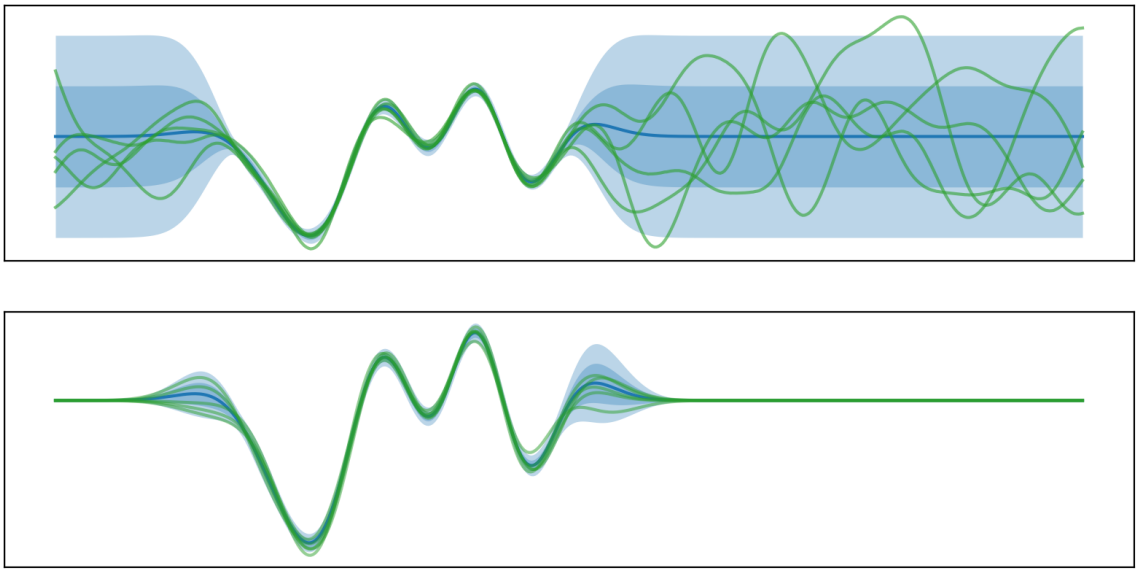


Fig. 2.3 Samples from the DTC (top) and SoR (bottom) posteriors. While both models have the same limited capacity to learn from data, DTC's posterior is a non-degenerate GP with infinite degrees of freedom.

We see this to be true by considering an arbitrarily large set of inputs X^* to predict on[3]. The predictive covariance will always be positive definite, as $K_{\mathbf{f}^*\mathbf{f}^*} - K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}^*}$ is the Schur complement of $K_{(*,\mathbf{u})(*,\mathbf{u})}$.

For this reason, we emphasise the following:

Remark. *The uncertainty of the DTC predictive distribution behaves like a non-parametric model, in the sense that sampled functions have infinite degrees of freedom. However, it can only be reduced in M degrees of freedom, so there is only limited capacity to* reduce *uncertainty in regions with lots of data.*

[3]We assume no repeated input points and $X^* \cap Z =$ to avoid technical details where two outputs are constrained to the same value, or have zero variance.

Additionally:

Remark. *The predictive mean of DTC behaves like a parametric model, in that it only has M degrees of freedom.*

Finally we want to emphasise a remark stated in Quiñonero-Candela and Rasmussen [2005]. They show that the joint prior distribution of $\mathbf{f}$ and $\mathbf{f}^*$ can be written as equation 2.11. While this is a perfectly valid Gaussian distribution, from which posteriors can be obtained, it does not correspond to exact inference in a modified Gaussian process, as the prior implies different marginal distributions for a training and testing point, even if the input location is identical.

$$p_{DTC}(\mathbf{f}, \mathbf{f}^*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}^*\end{bmatrix}; \begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}} & K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}^*}\\ K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}} & K_{\mathbf{f}^*\mathbf{f}^*}\end{bmatrix}\right) \tag{2.11}$$

Remark. *The DTC approximation does not correspond exactly to a Gaussian process.*

We also note an important distinction with the explicit basis function methods, is that the full GP solution is recovered if $M = N$ and all inducing inputs Z are at the input locations X. The explicit basis function methods, on the other hand, only recover the full GP solution in the infinite limit.

Selecting the inducing points In Seeger et al.'s [2003] original formulation of DTC, the set of inducing points was constrained to be a subset of the training inputs. Selecting a subset of M points from the N training points would be an intractable discrete optimisation problem. Instead training inputs were ranked by the KL divergence of the modified to the old posterior, and greedily added to the inducing points based on their ranking. Hyperparameters were optimised intermittently with adding inducing points using the model's marginal likelihood:

$$p_{DTC}(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}; 0, Q_{\mathbf{ff}} + \sigma^2 I\right), \tag{2.12}$$

$$Q_{\mathbf{ff}} = K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\,. \tag{2.13}$$

Note that the marginal likelihood is a function of the inducing inputs through $K_{\mathbf{fu}}$ and $K_{\mathbf{uu}}$, and so it can, in principle, be used for selecting the inducing inputs, similarly to Lázaro-Gredilla et al. [2010].

Desiderata DTC does not give a method for quantifying some distance to the exact solution. While the training method does imply a marginal likelihood (equation 2.12),

this may be an over- or under-estimate of the marginal likelihood of the full model. DTC does recover the exact posterior if the inducing inputs are chosen to be all the N training inputs. Additionally, its predictive error bars contain exactly the desirable property of non-parametric models.

2.3.5 Fully Independent Training Conditional (FITC) approximation

Snelson and Ghahramani [2006] propose a different approximate likelihood (equation 2.14). Again, Quiñonero-Candela and Rasmussen [2005] view this as a different training conditional. Compared to DTC's training conditional, FITC matches the variances of the exact conditional, but keeps the independence between training points required for computational tractability, as in equation 2.6.

$$p_{FITC}(\mathbf{y}|\mathbf{u}) = \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, \operatorname{diag}[K_{\mathbf{ff}} - Q_{\mathbf{ff}}] + \sigma^2 I\Big) \tag{2.14}$$

$$q(\mathbf{f}|\mathbf{u}) = \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, \operatorname{diag}[K_{\mathbf{ff}} - Q_{\mathbf{ff}}]\Big) \tag{2.15}$$

While DTC could be framed as learning using a finite basis function model and a regular Gaussian likelihood, things are more complicated for FITC. When viewing FITC as a likelihood approximation, it can be seen as learning a finite basis function model, with heteroskedastic (input dependent) noise in the likelihood [Snelson and Ghahramani, 2006]. Alternatively, in the unifying framework, the heteroskedastic noise is subsumed into the process. The posterior over $\mathbf{u}$ can then be seen as being obtained by performing exact inference in a modified GP with the kernel $\tilde{k}$:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \begin{cases} k(\mathbf{x}, \mathbf{x}'), & \text{if } \mathbf{x} = \mathbf{x}'. \\ k(\mathbf{x}, Z)k(Z, Z)^{-1}k(Z, \mathbf{x}) = \mathbf{k}_{f(\mathbf{x})\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}f(\mathbf{x})}, & \text{otherwise}. \end{cases} \tag{2.16}$$

If the testing and training conditionals are chosen to be the same, training and prediction correspond to exact inference with the kernel in equation 2.16 [Quiñonero-Candela and Rasmussen, 2005]. However, just like with DTC, the exact testing conditional can also be used, at the expense of the interpretation of exact inference in a GP.

The similarities with DTC mean that the same remarks concerning the non-parametric nature of the prediction applies to FITC. The same limitation also applies, in that the kernel mean only has M degrees of freedom. The main advantage of FITC over DTC, was that Snelson and Ghahramani [2006] showed that the inducing points

could be chosen using gradient based optimisation on FITC's marginal likelihood (equation 2.17), while this failed in DTC.

$$p_{FITC}(\mathbf{y}) = \mathcal{N}\Big(\mathbf{y}; 0, Q_{\mathbf{ff}} + \mathrm{diag}[K_{\mathbf{ff}} - Q_{\mathbf{ff}}] + \sigma^2 I\Big) \tag{2.17}$$

Desiderata Like DTC, FITC recovers the exact solution when $Z = X$. However, in chapter 3 we will show that when *optimising* the inducing points, FITC prefers to (sometimes significantly) depart from the full GP model. Additionally, there is no way of assessing how close to the exact solution FITC is – its marginal likelihood may be an over- or under-estimate of the true value. The error bars do maintain their non-parametric nature.

2.4 Inducing point posterior approximations

In the previous section we saw how modifications to the exact GP model could result in cheaper models that still maintained some of our desiderata. We also saw that the exact solution could be obtained by the approximate model with enough well-placed basis functions or inducing points. However, recovering the model does not give much reassurance about the quality of the approximation when resources are limited.

A more elegant way of approaching the problem is to attempt to directly approximate the exact posterior of the GP. We will discuss two well-established approximation techniques: Variational inference (VI) (see Blei et al. [2016] for a modern review) or Expectation Propagation (EP) [Minka, 2001]. Both VI and EP work by acknowledging that the true posterior is too complex or expensive to represent and deal with. Instead, the goal is to find the approximation in a restricted class of distributions which is tractable. EP and VI vary in how they choose the final posterior from this restricted class. Perhaps surprisingly, both methods result in posteriors which are strongly related to the model-based approximations from the previous sections.

2.4.1 A class of tractable Gaussian process posteriors

For parametric models, approximate posteriors can easily be parameterised using standard distributions. Gaussian process models are harder, as we need to represent distributions over functions. Luckily, as we saw earlier, we only need to handle the distributions over the finite dimensional marginals, and we will define the approximate posteriors in the same way.

In the previous sections we saw that DTC and FITC defined their predictive distributions using a posterior over inducing outputs $p(\mathbf{u}|\mathbf{y})$, and an testing conditional $p(\mathbf{f}|\mathbf{u})$, taken to be the prior's conditional distribution. The inducing outputs were integrated out in equation 2.10 to give a predictive distribution over any inputs. The marginals of the predictive distribution again defined a Gaussian process[4]. Taking inspiration from these methods, we can define a class of Gaussian processes with a very similar structure. Instead of committing to the distribution over $\mathbf{u}$ being a posterior, we free it to be *any* Gaussian. From the finite dimensional marginals, we can again find the implied Gaussian process:

$$\begin{aligned} q(\mathbf{f},\mathbf{u}) &= p(\mathbf{f}|\mathbf{u})q(\mathbf{u}) \\ &= \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big)\mathcal{N}(\mathbf{u};\boldsymbol{\mu},\Sigma)\,, \qquad (2.18) \\ q(\mathbf{f}) &= \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}, K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}}-\Sigma)K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big)\,, \qquad (2.19) \\ \implies f(\cdot) &\sim \mathcal{GP}\Big(k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}; k(\cdot,\cdot) - k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}}-\Sigma)K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\Big)\,. \qquad (2.20) \end{aligned}$$

These Gaussian processes can be seen as posteriors to alternate "fantasy" regression problems with M observations, inputs Z (controlling the covariances with $\mathbf{u}$) and an arbitrary likelihood[5]. To make this clearer, we consider the posterior of a GP where the inducing points $\mathbf{u}$ are observed, with an arbitrary Gaussian likelihood $\tilde{q}(\tilde{\mathbf{y}}|\mathbf{u})$, and its corresponding marginal likelihood $\tilde{q}(\tilde{\mathbf{y}})$:

$$q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})\frac{\tilde{q}(\tilde{\mathbf{y}}|\mathbf{u})p(\mathbf{u})}{\tilde{q}(\tilde{\mathbf{y}})}\,. \qquad (2.21)$$

The combination of the likelihood, the marginal likelihood and the prior will be $q(\mathbf{u})$, the posterior of the fictional regression problem. Since we allowed an arbitrary likelihood there will always exist be a Gaussian likelihood for any desired setting of $q(\mathbf{u}) = \mathcal{N}(\mathbf{u};\boldsymbol{\mu},\Sigma)$. The goal of the approximate inference scheme then becomes to adjust the "fantasy inputs" Z and the alternate likelihood using $\{\boldsymbol{\mu},\Sigma\}$, such that the posterior for this fantasy regression problem will be close to the true posterior, as in figure 2.1. These candidate posteriors have essentially the same properties as the DTC and FITC posterior. They can be manipulated with $\mathcal{O}(M^3)$ cost, are full rank, with error bars assuming infinite basis functions as in figure 2.3, but with only a finite number of basis functions in the mean, and so a limited capacity to adapt to data.

[4]We repeat the earlier point that the predictive Gaussian process did *not* need to be consistent with the Gaussian process used for training $\mathbf{u}$.

[5]Thanks to Richard Turner for popularising this view in the group.

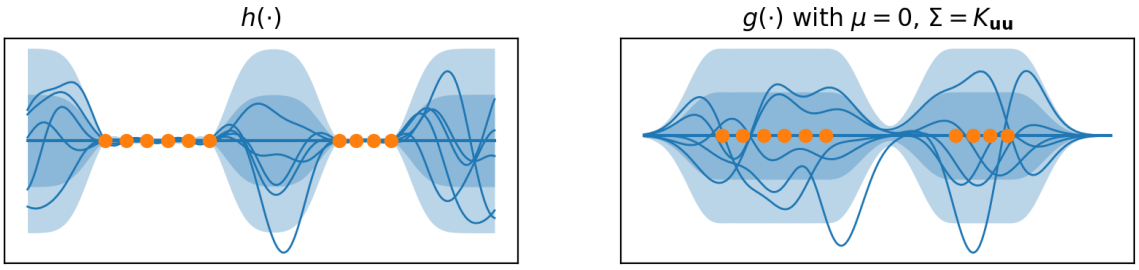


Fig. 2.4 The exact GP prior split into its components $h(\cdot)$ and $g(\cdot)$. $h(\cdot)$ remains unchanged after observing data, while $g(\cdot)$ can be adjusted.

We can probe the class' properties a bit deeper by observing that the approximate Gaussian process posterior is the sum of two Gaussian processes (noted similarly in Hensman et al. [2018]):

$$f(\cdot) = g(\cdot) + h(\cdot)\,, \tag{2.22}$$

$$g(\cdot) \sim \mathcal{GP}\left(k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}; k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\Sigma K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\right), \tag{2.23}$$

$$h(\cdot) \sim \mathcal{GP}\left(0, k(\cdot,\cdot) - k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\right). \tag{2.24}$$

For some choice of inducing input locations, the component $g(\cdot)$ contains the degrees of freedom that *can* be learned about by the approximate posterior. It has M basis functions to adjust in the mean. Additionally, if there is any uncertainty left in the value of the weights, Σ can increase the variance. In the extreme case of $\Sigma = 0$, the posterior has reached its capacity to learn, while for $\Sigma = K_{\mathbf{uu}}$, $f(\cdot)$ will simply be the prior. The component $h(\cdot)$, on the other hand, represents all degrees of freedom in the prior that the approximate posterior *cannot* learn about. This provides the error bars of the non-parametric prior.

The split is illustrated in figure 2.4. $h(\cdot)$ shows what variance is left if the inducing points are fully known, while $g(\cdot)$ shows what degrees of freedom from the prior can be learned about. Figure 2.5 shows what happens when $g(\cdot)$ is constrained after observing some data. We see that the variance of the process can be reduced by observations near inducing points, but not ones far from inducing points. The distribution of the inducing outputs far from the data stays near the prior, and does not help modelling the data. This illustrates the crucial role of placing the inducing points when attempting to find a good approximation.

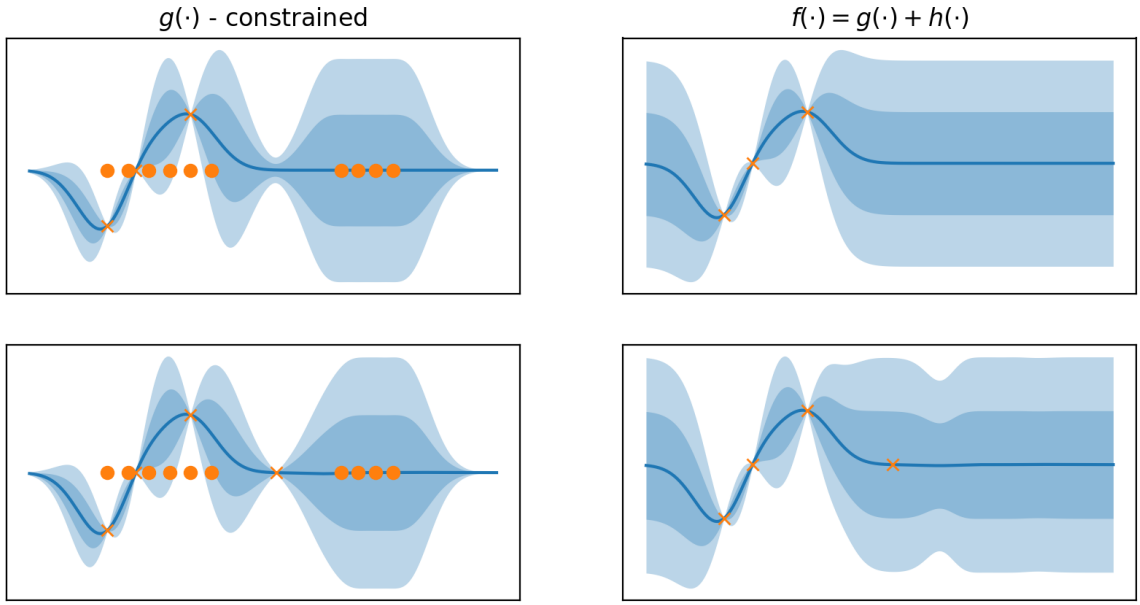


Fig. 2.5 Top: $g(\cdot)$ after observing some data, and the process resulting from the combination. Bottom: Idem, but with an input far from any inducing points. The process cannot represent the desired reduction in variance, as the $h(\cdot)$ component remains unchanged.

2.4.2 Variational inference

The main idea behind variational inference is to restrict the approximate posterior to a class of tractable distributions, and choose the one with the lowest KL divergence to the true posterior. Computing the KL divergence to the true posterior is intractable in general, since it requires the quantity we want to approximate in the first place. Instead, a lower bound (denoted $\mathcal{L}$) to the marginal likelihood can be constructed, where the slack is exactly equal to the intractable KL divergence. We can derive the bound simply by applying Bayes rule after stating the KL divergence. Here we use $q(\mathcal{H})$ to denote the approximate posterior over the hidden variables, which will be chosen from the tractable constrained family, while $\mathcal{D}$ denotes the observed data.

$$\begin{aligned}\mathrm{KL}[\, q(\mathcal{H}) \,\|\, p(\mathcal{H}|\mathcal{D}) \,] &= \mathbb{E}_{q(\mathcal{H})}\left[\log \frac{q(\mathcal{H})}{p(\mathcal{H}|\mathcal{D})}\right] = \mathbb{E}_{q(\mathcal{H})}\left[\log \frac{q(\mathcal{H})p(\mathcal{D})}{p(\mathcal{D}|\mathcal{H})p(\mathcal{H})}\right] \\ &= \log p(\mathcal{D}) - \mathbb{E}_{q(\mathcal{H})}[\log p(\mathcal{D}|\mathcal{H}) + \log p(\mathcal{H}) - \log q(\mathcal{H})] \qquad (2.25)\end{aligned}$$

Since the marginal likelihood is constant w.r.t. $q(\mathcal{H})$, the expectations form a lower bound to it, where the gap is exactly the KL divergence we want to minimise:

$$\mathcal{L} \stackrel{\text{def}}{=} \mathbb{E}_{q(\mathcal{H})}[\log p(\mathcal{D}|\mathcal{H}) + \log p(\mathcal{H}) - \log q(\mathcal{H})] \tag{2.26}$$

$$\mathcal{L} = \log p(\mathcal{D}) - \mathrm{KL}[\, q(\mathcal{H}) \,\|\, p(\mathcal{H}|\mathcal{D}) \,] \tag{2.27}$$

$$\therefore \mathcal{L} \leq \log p(\mathcal{D}) \tag{2.28}$$

For some models the optimal $q(\mathcal{H})$ can be found in closed form using calculus of variations (e.g. Beal [2003]), for others $\mathcal{L}$ is calculated in closed form and $q(\mathcal{H})$ is parameterised and optimised using gradients. More recently, it has been shown [Titsias and Lázaro-Gredilla, 2014; Ranganath et al., 2014] that low-variance unbiased estimates of $\mathcal{L}$ can be used if a closed form expression is hard to come by.

Variational inference for Gaussian processes

Titsias [2009b] introduced a method relying on the variational lower bound to find the approximate posterior. In light of Matthews et al.'s [2016] recent work showing that maximising that variational bound did, in fact, minimise the KL divergence between the prior and posterior stochastic process, we will present the informal derivation from the latter work. We will refer to this approximation as the Variational Free Energy (VFE) approximation.

We start with a Gaussian process prior over the latent function, and a factorised likelihood that only depends on the Gaussian process at the locations X:

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))\,, \tag{2.29}$$

$$p(\mathbf{y}|f(X)) = \prod_{n=1}^{N} p(y_n|f(\mathbf{x}_n))\,. \tag{2.30}$$

As we saw in section 1.2.3, our posterior process only requires inference about the function values at the input locations X, and so only needs the marginal prior of $f(X)$. However, we are free to consider the marginal over a larger set of variables – we are simply representing more of the same latent Gaussian process. In order to emphasise that we are inferring the entire process, we will consider the extra variables $\mathbf{u} \stackrel{\text{def}}{=} f(Z)$ and $\mathbf{f}^* \stackrel{\text{def}}{=} f(X^*)$. For now, the inputs Z and X^* will be completely arbitrary in size and location. Following Bayes' rule and using the product rule to split the joint prior

into its conditionals, we get the posterior over the finite dimensional marginal:

$$p(\mathbf{f}, \mathbf{u}, \mathbf{f}^*|\mathbf{y}) = \frac{\prod_{n=1}^{N} p(y_n|\mathrm{f}_n)p(\mathbf{f}|\mathbf{u})p(\mathbf{u})}{p(\mathbf{y})} p(\mathbf{f}^*|\mathbf{u}, \mathbf{f})\,. \tag{2.31}$$

Next, we choose our class of approximating Gaussian processes to be the one described in section 2.4.1. As described, it can be defined by its finite dimensional distribution over the inducing inputs Z ($q(\mathbf{u})$) and the inducing conditional $p(\mathbf{f}^*, \mathbf{f}|\mathbf{u})$). We will represent the approximating Gaussian process at the same inputs X, X^* and Z.

We can now follow the same variational inference procedure as described earlier, by minimising the KL divergence between the two finite dimensional marginals. We note that the finite dimensional marginal can be factorised in similar ways:

$$\mathrm{KL}[\, q(\mathbf{f}, \mathbf{u}, \mathbf{f}^*) \,\|\, p(\mathbf{f}, \mathbf{u}, \mathbf{f}^*|\mathbf{y}) \,] = \mathrm{KL}[\, q(\mathbf{u})p(\mathbf{f}|\mathbf{u})p(\mathbf{f}^*|\mathbf{f}, \mathbf{u}) \,\|\, p(\mathbf{f}, \mathbf{u}|\mathbf{y})p(\mathbf{f}^*|\mathbf{f}, \mathbf{u}) \,]\,. \tag{2.32}$$

By considering the actual integral formulation of the KL, we can see that the common factor $p(\mathbf{f}^*|\mathbf{f}, \mathbf{u})$ cancels out, and *does not influence the value of the KL*, i.e.:

$$\mathrm{KL}[\, q(\mathbf{f}, \mathbf{u}, \mathbf{f}^*) \,\|\, p(\mathbf{f}, \mathbf{u}, \mathbf{f}^*|\mathbf{y}) \,] = \mathrm{KL}[\, q(\mathbf{u})p(\mathbf{f}|\mathbf{u}) \,\|\, p(\mathbf{f}, \mathbf{u}|\mathbf{y}) \,]\,. \tag{2.33}$$

Regardless of how many extra points X^* we evaluate the KL divergence on, the value will be equal to considering the finite dimensional marginals only. In fact, Matthews et al. [2016] showed this expression to be equal to a formal version of a KL divergence between the approximate and posterior processes.

Now that we are satisfied with the properties of the KL divergence, we can set about to minimise it by constructing a lower bound, as introduced by Titsias [2009b]. The fact that both the approximate and true process contain $p(\mathbf{f}|\mathbf{u})$ – the expensive term with an $\mathcal{O}(N^3)$ matrix operation – allows them to cancel, which is the essence of the trick introduced by Titsias:

$$\mathrm{KL}[\, q(\mathbf{u})p(\mathbf{f}|\mathbf{u}) \,\|\, p(\mathbf{f}, \mathbf{u}|\mathbf{y}) \,] = \log p(\mathbf{y}) - \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{u})}\left[\log \frac{p(\mathbf{y}|\mathbf{f})\cancel{p(\mathbf{f}|\mathbf{u})}p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})}q(\mathbf{u})}\right]}_{=\mathcal{L}}, \tag{2.34}$$

$$\therefore \mathcal{L} = \sum_{n=1}^{N} \int p(f_n|\mathbf{u})q(\mathbf{u}) \log p(y_n|f_n) \mathrm{d}f_n \mathrm{d}\mathbf{u} - \mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u}) \,]\,. \tag{2.35}$$

We can integrate out $\mathbf{u}$ further to show that the likelihood expectation depends only on the marginal distribution of the approximating process at $\mathbf{x}_n$:

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n|f(\mathbf{x}_n)] - \mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u})\,]\,. \tag{2.36}$$

The above expression works for general likelihoods that depend point-wise on the latent function [Hensman et al., 2015b,a]. It also lends itself to stochastic optimisation using minibatches by subsampling the sum over N data points [Hensman et al., 2013]. For regression, Titsias [2009b] showed that $q(\mathbf{u})$ could optimised in free form, resulting in a Gaussian distribution. In turn, this allowed the explicit representation of $q(\mathbf{u})$ to be replaced by the optimised form, resulting in the so-called collapsed bound:

$$\mathcal{L} = \mathcal{N}\Big(\mathbf{y}; 0, K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}} + \sigma^2 I\Big) - \frac{1}{2\sigma^2}\,\mathrm{Tr}\Big(K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big)\,. \tag{2.37}$$

The lower bound can only improve with more inducing points (see chapter 3 or Matthews [2016] for separate proofs).

An upper bound Usually, tractable upper bounds to the marginal likelihood are not known, and variational inference does not help in finding them. However, in the special case of Gaussian process regression, Titsias [2014] did construct the upper bound:

$$\begin{aligned}\log p(\mathbf{y}) \leq &-\frac{N}{2}\log 2\pi - \frac{1}{2}\log\Big|K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}} + \sigma^2 I\Big| \\ &- \frac{1}{2}\mathbf{y}^{\mathsf{T}}\Big(K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}} + (c+\sigma^2)I\Big)^{-1}\mathbf{y}\,, \qquad (2.38)\\ c = &\,\mathrm{Tr}\Big(K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big) \geq \lambda_{max}\Big(K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big)\,. \qquad (2.39)\end{aligned}$$

The proof relies on observing that $|K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}| \leq |K_{\mathbf{ff}}|$ (the normalising constant of $p(\mathbf{y})$), while ensuring that c compensates for the smaller eigenvalues of $K_{\mathbf{fu}}K_{\mathbf{uu}}K_{\mathbf{uf}}$ such that the quadratic term is over-estimated. c can be taken as any upper bound to the eigenvalues of $K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}$. With application of the Woodbury matrix identity, this upper bound is also computable in $\mathcal{O}(NM^2)$. Results on a toy dataset in Titsias [2014] show that the upper bound converges to the marginal likelihood slower than the lower bound, with increasing M.

Finding tighter upper bounds would be very useful for sandwiching marginal likelihoods. Especially finding tighter bounds on c could help, as it is a rather drastic

change to the quadratic term. One other bound, which may be tighter in certain situations, is the maximum row sum of a matrix (from the Perron-Frobenius theorem).

Desiderata Like FITC and DTC, VFE fully maintains the non-parametric error bars (in fact, its posterior is equivalent to that of DTC Titsias [2009b]), and recovers the true results when $Z = X$. It is also guaranteed to recover the true posterior when globally optimising Z, given that it minimises the KL divergence, and that it monotonically improves with more resources (see chapter 3). However, local optima in the objective function may cause sub-optimal solutions to be found with different initialisations.

Contrary to DTC and FITC, the lower bound to the marginal likelihood provides some way to assess the quality of the approximation. While knowledge of the KL divergence between the posterior and approximation would be ideal, the lower bound can guide when to add more capacity to the approximation: a halt in the increase of the bound with increasing M can indicate that the bound has become tight. Stronger evidence for this can be gained by using the upper bound to sandwich the marginal likelihood. A small gap between upper and lower bound guarantees an accurate approximation.

2.4.3 Expectation propagation

Expectation Propagation (EP) [Minka, 2001] is an alternative method for approximate inference. Rather than directly minimising an objective function, it iteratively refines a posterior by minimising the KL divergence of marginals of a tractable posterior with the *tilted* distribution. The tilted distribution is the approximate posterior with one intractable, but exact, likelihood term added in.

Expectation propagation has been applied to general graphical models [Minka et al., 2014], but also non-sparse Gaussian processes with non-Gaussian likelihoods (see [Rasmussen and Williams, 2005; Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008, §3.6]). To obtain sparse inference algorithms with non-Gaussian likelihoods, e.g. Naish-Guzman and Holden [2008]; Hernández-Lobato and Hernández-Lobato [2016] start from FITC as an approximate model, and use EP for the approximate likelihoods.

EP gives rise to FITC For the purposes of this thesis, we will not delve into details of EP, as Bui et al. [2017] provide an excellent overview of EP, its variants, and their close relationship to the variational method described above. However, the connection between EP and FITC that was brought to light is critical to the points

raised in this thesis: Performing EP on the full Gaussian process regression model, using the same approximate posterior as used in the variational free energy (VFE) method (section 2.4.1), results in the same solution as FITC. Additionally, adding non-Gaussian likelihoods, leads to the approximations cited above. Bui et al. [2017] also point out that a similar derivation was already developed by Csató and Opper [2002], and the link between EP and FITC was mentioned in Qi et al. [2010], without these links becoming widely known.

For the purposes of this thesis, we will analyse FITC as a model in itself. However, given that it is in fact the same method as an EP approximation would result in, FITC can equally be presented as an EP approximation.

2.5 Conclusion

We have discussed 5 common approximation schemes for Gaussian processes: Random Fourier features, SSGP, DTC, FITC (equivalent to EP) and VFE, and discussed some of their properties against the desiderata for non-parametric approximations laid out in section 2.1. We list a summary in table 2.1. We saw that it was possible to maintain the non-parametric error bars in prediction, even in the approximate models.

All methods had to give up the consistency property of non-parametric models. All methods except VFE and the EP interpretation of FITC propose approximate models. It is interesting to note that all "approximate model" methods (SSGP, DTC and FITC) lose their ability to recover the true model, even when given enough resources to do so (see chapter 3), if the inducing inputs are optimised for a more compressed representation. Only Random Fourier features and VFE seem to have guarantees and assessment criteria which show that the approximation will ever more closely resemble the target model. VFE seems to be the only method to exhibit all desirable properties. It only gives up the infinite capacity of the non-parametric model, but retains its error bars, recovers the exact solution *and* has a computable method to assess how far away it is in terms of the lower (and upper) bound.

For the remainder of this thesis, we focus on methods that maintain the non-parametric error bars, particularly FITC and VFE. While we do not want to detract from the usefulness of Random Fourier features, it is interesting to push approximations which do not compromise on such a fundamental property of the non-parametric GP that is to be approximated.

The properties discussed so far are only theoretical, and the main aim is to obtain a good approximation for some constrained M. So one could argue that the desiderata

	Assessment	Recovery	Monotone improvement	Error bars
RFF	✓[1]	✓	✓[2]	×
SSGP	×	×	×	×
DTC	×	?[3]	×	✓
FITC	×	?[3]	×	✓
VFE	✓	✓	✓	✓

[1] Through a probabilistic bound on the deviation of the effective kernel in a small region, rather than a global guarantee.
[2] In the sense of its kernel deviation.
[3] Only if the inducing points are chosen from the inputs.

Table 2.1 Summary of GP approximation methods against the desiderata laid out in section 2.1.

of assessment, recovery and monotone improvement are not strictly necessary. In the next chapter, we delve into the detailed behaviour of FITC and VFE to assess the relative qualities of their approximations.

Chapter 3

Understanding the behaviour of FITC and VFE

In the previous section, we introduced two sparse approximations and showed how two objective functions – FITC and VFE – for learning sparse approximations could be derived. While both methods have been around for several years, there still is not a consensus in the literature on which method is "best". The original papers [Snelson and Ghahramani, 2006; Titsias, 2009b] show some illustrative 1D examples, together with benchmarks on some standard datasets. There has been successful follow-on work extending both methods [Naish-Guzman and Holden, 2008; Titsias and Lawrence, 2010; Damianou and Lawrence, 2013], without much clarity on the detailed properties of each method, and when either method is appropriate to use. A certain amount of "folk-wisdom" has developed among practitioners over the years, with certain (sometimes contradictory) behaviours mentioned in different papers.

In this work, we aim to thoroughly investigate and characterise the difference in behaviour of the FITC and VFE approximations, when applied to regression. We analyse each method and shed light on their behaviours and when they can be observed, hopefully resolving some of the contradictory claims from the past. We do this by unpicking the methods' optimisation behaviour from the biases of their objective functions. Our aim is to understand the approximations in detail in order to know under which conditions each method is likely to succeed or fail in practice. We will assess both methods on how they approximate the true GP model based on the desiderata outlined in the previous chapter, rather than placing the emphasis on predictive performance.

Additionally, in section 3.3, we discuss the curiosity that FITC and VFE share a variational lower bound. This highlights another interesting link between the methods,

which is perhaps worrying given that VFE was designed to approximate the *full* GP model, not an approximation to it. This connection has been used to attempt to develop variational approximations for heteroscedastic models. We show that this approach may have unintended consequences.

3.1 Common notation

We saw the optimisation objectives for VFE and FITC in equation 2.37 and equation 2.17 respectively. We will repeat them here in a common notation:

$$\mathcal{F} = \frac{N}{2}\log(2\pi) + \underbrace{\frac{1}{2}\log|Q_{\mathbf{ff}} + G|}_{\text{complexity penalty}} + \underbrace{\frac{1}{2}\mathbf{y}^{\mathsf{T}}(Q_{\mathbf{ff}} + G)^{-1}\mathbf{y}}_{\text{data fit}} + \underbrace{\frac{1}{2\sigma_n^2}\operatorname{Tr}(T)}_{\text{trace term}}, \tag{3.1}$$

where

$$G_{\text{FITC}} = \operatorname{diag}[K_{\mathbf{ff}} - Q_{\mathbf{ff}}] + \sigma_n^2 I \qquad G_{\text{VFE}} = \sigma_n^2 I \tag{3.2}$$

$$T_{\text{FITC}} = 0 \qquad T_{\text{VFE}} = K_{\mathbf{ff}} - Q_{\mathbf{ff}}. \tag{3.3}$$

The common objective function has three terms, of which the data fit and complexity penalty have direct analogues to the full GP. The **data fit** term penalises the data lying outside the covariance ellipse $Q_{\mathbf{ff}} + G$. The **complexity penalty** is the integral of the data fit term over all possible observations $\mathbf{y}$. It characterises the *volume* of possible datasets that are compatible with the data fit term. This can be seen as the mechanism of *Occam's razor* [Rasmussen and Ghahramani, 2001], by penalising the methods for being able to predict too many datasets. The **trace term** in VFE ensures that the objective function is a true lower bound to the marginal likelihood of the full GP. Without this term, VFE is identical to the earlier DTC approximation [Seeger et al., 2003] which can over-estimate the marginal likelihood. The trace term penalises the sum of the conditional variances at the training inputs, conditioned on the inducing inputs [Titsias, 2009a]. Intuitively, it ensures that VFE not only models this specific dataset $\mathbf{y}$ well, but also approximates the covariance structure of the full GP $K_{\mathbf{ff}}$.

3.2 Comparative behaviour

As our main test case we use the same one dimensional dataset considered throughout this thesis and in Snelson and Ghahramani [2006] & Titsias [2009b]. Of course, sparse methods are not necessary for this toy problem, but all of the issues we raise

are illustrated nicely in this one dimensional task which can easily be plotted. In sections 3.2.1 to 3.2.3 we illustrate issues relating to the *objective functions*. These properties are independent of how the method is optimised, and should generalise to stochastic optimisation methods that use the same bound [Hensman et al., 2013]. However, whether they are encountered in practice depends on the optimiser dynamics. We describe this influence in sections 3.2.4 and 3.2.5.

This section is joint work with Matthias Bauer and was published in Bauer, van der Wilk, and Rasmussen [2016]. The authors agree that the paper was the result of equal contribution, with the experiments using the upper-bound being done by Mark van der Wilk alone. The supplementary material of the paper included a first-order analysis of repeated inducing points with jitter led by Matthias Bauer, which is not included in this thesis.

3.2.1 FITC can severely underestimate the noise variance, VFE overestimates it

In the full GP with a Gaussian likelihood we assume a homoscedastic (input *in*dependent) noise model with noise variance parameter σ_n^2. It fully characterises the uncertainty left after completely learning the latent function. Here, we show how FITC can also use the diagonal term $\operatorname{diag}(K_{\mathbf{ff}} - Q_{\mathbf{ff}})$ in G_{FITC} as heteroscedastic (input dependent) noise [Snelson and Ghahramani, 2006] to account for these differences, thus, invalidating the above interpretation of the noise variance parameter. In fact, the FITC objective function encourages underestimation of the noise variance, whereas the VFE bound encourages overestimation. The latter is in line with previously reported biases of variational methods [Turner and Sahani, 2011].

Figure 3.1 shows the configuration most preferred by the FITC objective for a subset of 100 data points of the Snelson dataset, found by an exhaustive systematic search for a minimum over hyperparameters and inducing inputs locations. The procedure relied on several random re-starts of the joint optimisation of Z and hyperparameters, choosing the solution with the highest marginal likelihood. Next, inducing points were incrementally added in locations that would increase the marginal likelihood most. For this solution, the noise variance is shrunk to practically zero, despite the mean prediction not going through every data point. Note how the mean still behaves well and how the training data lie well within the predictive variance. Only when considering predictive probabilities will this behaviour cause diminished performance.

VFE, on the other hand, is able to approximate the posterior predictive distribution almost exactly.

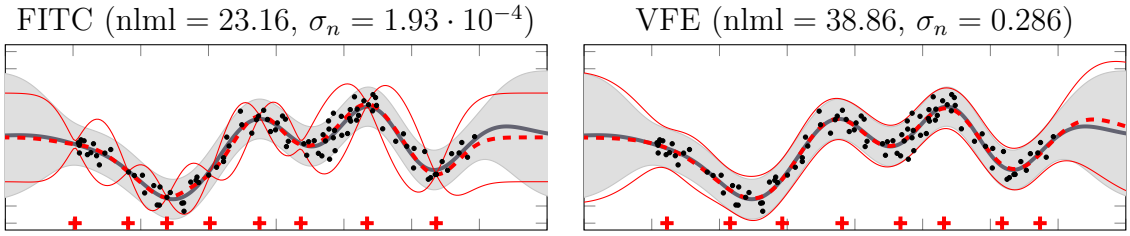


Fig. 3.1 Behaviour of FITC and VFE on a subset of 100 data points of the Snelson dataset for 8 inducing inputs (red crosses indicate inducing inputs; red lines indicate mean and 2σ) compared to the prediction of the full GP in grey. Optimised values for the full GP: NLML = 34.15, $\sigma_n = 0.274$.

For both approximations, the complexity penalty decreases with decreased noise variance, by reducing the volume of datasets that can be explained. For a full GP and VFE this is accompanied by a data fit penalty for data points lying far away from the predictive mean. FITC, on the other hand, has an additional mechanism to avoid this penalty: its diagonal correction term $\mathrm{diag}(K_{\mathbf{ff}} - Q_{\mathbf{ff}})$. This term can be seen as an input dependent or heteroscedastic noise term (discussed as a modelling advantage by Snelson and Ghahramani [2006]), which is zero exactly at an inducing input, and which grows to the prior variance away from an inducing input. By placing the inducing inputs near training data that happen to lie near the mean, the heteroscedastic noise term is locally shrunk, resulting in a reduced complexity penalty. Data points both far from the mean and far from inducing inputs do not incur a data fit penalty, as the heteroscedastic noise term has increased around these points. This mechanism removes the need for the homoscedastic noise to explain deviations from the mean, such that σ_n^2 can be turned down to reduce the complexity penalty further.

This explains the extreme pinching (severely reduced noise variance) observed in figure 3.1, also see, e.g. Titsias [2009b, Fig. 2]. In examples with more densely packed data, there may not be any places where a near-zero noise point can be placed without incurring a huge data-fit penalty. However, inducing inputs will be placed in places where the data happens to randomly cluster around the mean, which still results in a decreased noise estimate, albeit less extreme, see figures 3.2 and 3.3 where we use all 200 data points.

Remark 1. *FITC has an alternative mechanism to explain deviations from the learned function than the likelihood noise and will underestimate σ_n^2 as a consequence. In extreme cases, σ_n^2 can incorrectly be estimated to be almost zero.*

As a consequence of this additional mechanism, σ_n^2 can no longer be interpreted in the same way as for VFE or the full GP. σ_n^2 is often interpreted as the amount of uncertainty in the dataset which cannot be explained, regardless of future data. Based on this interpretation, a low σ_n^2 is often used as an indication that the dataset is being fitted well. Active learning applications rely on a similar interpretation to differentiate between inherent noise, and uncertainty that can be reduced with more data. FITC's different interpretation of σ_n^2 will cause efforts like these to fail.

VFE, on the other hand, is biased towards over-estimating the noise variance, because of both the data fit and the trace term. $Q_{\mathbf{ff}} + \sigma_n^2 I$ has $N - M$ eigenvectors with an eigenvalue of σ_n^2, since the rank of $Q_{\mathbf{ff}}$ is M. Any component of $\mathbf{y}$ in these directions will result in a larger data fit penalty than for $K_{\mathbf{ff}}$, which can only be reduced by increasing σ_n^2. This is actually likely to be the case since the number of basis functions is limited. The trace term can also be reduced by increasing σ_n^2.

Remark 2. *The VFE objective tends to over-estimate the noise variance compared to the full GP.*

3.2.2 VFE improves with additional inducing inputs, FITC may ignore them

Here we investigate the behaviour of each method when more inducing inputs are added. For both methods, adding an extra inducing input gives it an extra basis function to model the data with. We discuss how and why VFE always improves, while FITC may deteriorate.

Figure 3.2 shows an example of how the objective function changes when an inducing input is added anywhere in the input domain. While the change in objective function looks reasonably smooth overall, there are pronounced spikes for both FITC and VFE. These return the objective to the value without the additional inducing input and occur at the locations of existing inducing inputs. We discuss the general change first before explaining the spikes.

Mathematically, adding an inducing input corresponds to a rank 1 update of $Q_{\mathbf{ff}}$, and can be shown to always improve VFE's bound[1], see appendix B for a proof. VFE's complexity penalty increases due to an extra non-zero eigenvalue in $Q_{\mathbf{ff}}$, but gains in data fit and trace.

[1]Matthews [2016] independently proved this result by considering the KL divergence between processes. Titsias [2009b] proved this result for the special case when the new inducing input is selected from the training data.

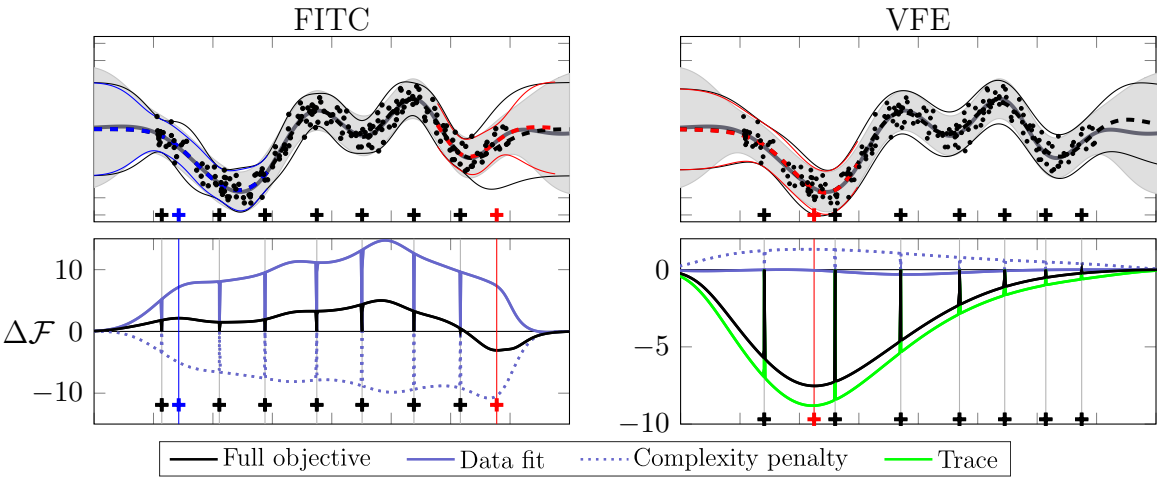


Fig. 3.2 *Top:* Fits for FITC and VFE on 200 data points of the Snelson dataset for $M = 7$ optimised inducing inputs (black). *Bottom:* Change in objective function from adding an inducing input anywhere along the x-axis (no further hyperparameter optimisation performed). The overall change is decomposed into the change in the individual terms (see legend). Two particular additional inducing inputs and their effect on the predictive distribution shown in red and blue.

Remark 3. *VFE's posterior and marginal likelihood approximation become more accurate (or remain unchanged) regardless of where a new inducing input is placed.*

For FITC, the objective can change either way. Regardless of the change in objective, the heteroscedastic noise is decreased at all points (see appendix B for the proof). For a squared exponential kernel, the decrease is strongest around the newly placed inducing input. This decrease has two effects. Firstly, it reduces the complexity penalty since the diagonal component of $Q_{\mathbf{ff}} + G$ is reduced and replaced by a more strongly correlated $Q_{\mathbf{ff}}$. Secondly, it worsens the data fit term since the heteroscedastic term is relied upon to fit the data when the homoscedastic noise is underestimated. Figure 3.2 shows reduced error bars with several data points now outside of the 95% prediction bars. Also shown is a case where an additional inducing input improves the objective, where the extra correlations outweigh the reduced heteroscedastic noise.

Both VFE and FITC exhibit pathological behaviour (spikes) when inducing inputs are clumped, that is, when they are placed exactly on top of each other. In this case, the objective function has the same value as when all duplicate inducing inputs were removed. In other words, for all practical purposes, a model with duplicate inducing inputs reduces to a model with fewer, individually placed inducing inputs.

Theoretically, these pathologies only occur at single points, such that no gradients towards or away from them could exist and they would never be encountered. In practise, however, these peaks are widend by a finite *jitter* that is added to $K_{\mathbf{uu}}$ to

ensure it remains well conditioned enough to be invertible. This finite width provides the gradients that allow an optimiser to detect these configurations.

As VFE always improves with additional inducing inputs, these configurations must correspond to maxima of the optimisation surface and clumping of inducing inputs does not occur for VFE. For FITC, configurations with clumped inducing inputs can – and often do – correspond to minima of the optimisation surface. By placing inducing inputs on top of each other, FITC can avoid the penalty of adding an extra inducing input and can gain the bonus from the heteroscedastic noise. Clumping, thus, constitutes a mechanism that allows FITC to effectively remove inducing inputs at no cost. In practise we find that convergence towards these minima can be slow. Titsias [2009b] notes a variation of this effect when describing that FITC often does not spread the inducing points around the data to optimally benefit from the added basis functions.

We illustrate this behaviour in figure 3.3 for 15 randomly initialised inducing inputs. FITC places some of them exactly on top of each other, whereas VFE spreads them out and recovers the full GP well.

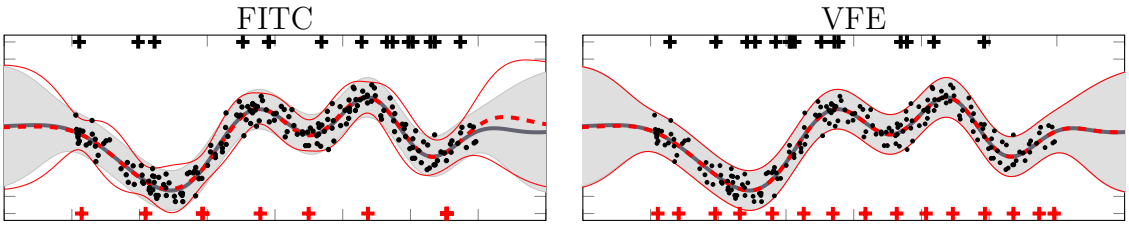


Fig. 3.3 Fits for 15 inducing inputs for FITC and VFE (initial as black crosses, optimised red crosses). Even following joint optimisation of inducing inputs and hyperparameters, FITC avoids the penalty of added inducing inputs by clumping some of them on top of each other (shown as a single red cross). VFE spreads out the inducing inputs to get closer to the true full GP posterior.

Remark 4. *In FITC, having a good approximation $Q_{\mathbf{ff}}$ to $K_{\mathbf{ff}}$ needs to be traded off with the gains coming from the heteroscedastic noise. FITC does not always favour a more accurate approximation to the GP.*

Remark 5. *FITC avoids losing the gains of the heteroscedastic noise by placing inducing inputs on top of each other, effectively removing them.*

3.2.3 FITC does not recover the true posterior, VFE does

In the previous section we showed that FITC has trouble using additional resources to model the data, and that the optimiser moves away from a more accurate approximation

to the full GP. Here we show that this behaviour is inherent to the FITC objective function.

Both VFE and FITC *can* recover the true posterior by placing an inducing input on every training input [Titsias, 2009b; Snelson, 2007]. For VFE, this must be a *global* minimum, since the KL gap to the true marginal likelihood is zero. For FITC, however, this solution is not stable and the objective can still grow by aggressive optimisation, as was shown in Matthews [2016]. The derivative of the inducing inputs is zero for this configuration, but the objective function can still be improved. As with the clumping behaviour, adding jitter subtly makes this behaviour more obvious by perturbing the gradients. In figure 3.4 we show this behaviour on a subset of 100 data points of the Snelson dataset. VFE is at a minimum and does not move the inducing inputs, whereas FITC improves its objective and clumps the inducing inputs considerably.

Method	**NLML initial**	**nlml optimised**
Full GP	-	33.8923
VFE	33.8923	33.8923
FITC	33.8923	28.3869

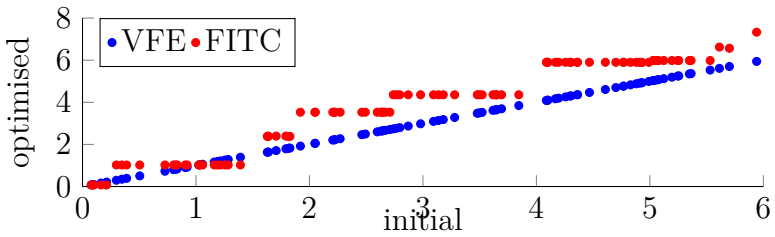


Fig. 3.4 Results of optimising VFE and FITC after initialising at the solution that gives the correct posterior and marginal likelihood. We observe that FITC moves to a significantly different solution with better objective value (Table, *top*) and clumped inducing inputs (Figure, *bottom*).

Remark 6. *FITC generally does not recover the full GP, even when it has enough resources.*

3.2.4 FITC relies on local optima

So far, we have observed some cases where FITC fails to produce results in line with the full GP, and characterised why. However, in practice, FITC has performed well, and pathological behaviour is not always observed. In this section we discuss the optimiser dynamics and show that they help FITC behave reasonably.

To demonstrate this behaviour, we consider a 4d toy dataset: 1024 training and 1024 test samples drawn from a 4d Gaussian Process with isotropic squared exponential covariance function ($l = 1.5, s_f = 1$) and true noise variance $\sigma_n^2 = 0.01$. The data inputs were drawn from a Gaussian centred around the origin, but similar results were obtained for uniformly sampled inputs. We fit both FITC and VFE to this dataset with the number of inducing inputs ranging from 16 to 1024, and compare a representative run to the full GP in figure 3.5.

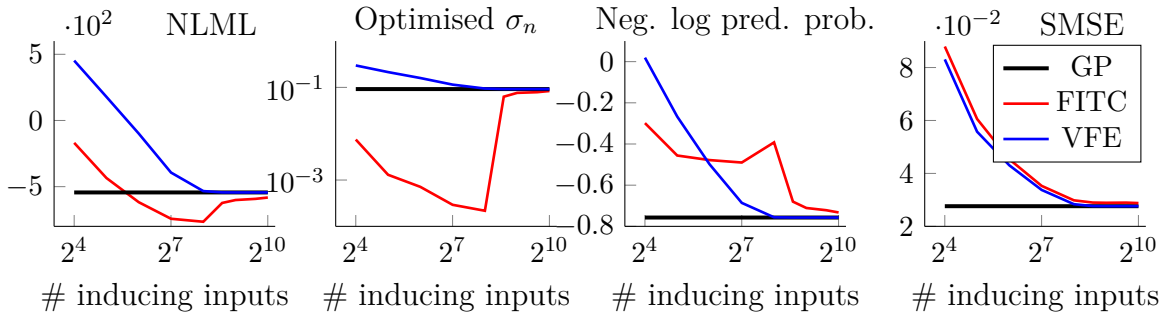


Fig. 3.5 Optimisation behaviour of VFE and FITC for varying number of inducing inputs compared to the full GP. We show the objective function (negative log marginal likelihood), the optimised noise σ_n, the negative log predictive probability and standardised mean squared error as defined in Rasmussen and Williams [2005].

VFE monotonically approaches the values of the full GP but initially overestimates the noise variance, as discussed in section 3.2.1. Conversely, we can identify three regimes for the objective function of FITC: 1) Monotonic improvement for few inducing inputs, 2) a region where FITC over-estimates the marginal likelihood, and 3) recovery towards the full GP for many inducing inputs. Predictive performance follows a similar trend, first improving, then declining while the bound is estimated to be too high, followed by a recovery. The recovery is counter to the usual intuition that over-fitting worsens when adding more parameters.

We explain the behaviour in these three regimes as follows: When the number of inducing inputs are severely limited (regime 1), FITC needs to place them such that $K_{\mathbf{ff}}$ is well approximated. This correlates most points to some degree, and ensures a reasonable data fit term. The marginal likelihood is under-estimated due to lack of a flexibility in $Q_{\mathbf{ff}}$. This behaviour is consistent with the intuition that limiting model capacity prevents overfitting.

As the number of inducing inputs increases (regime 2), the marginal likelihood is over-estimated and the noise drastically under-estimated. Additionally, performance in terms of log predictive probability deteriorates. This is the regime closest to FITC's

behaviour in figure 3.1. There are enough inducing inputs such that they can be placed such that a bonus can be gained from the heteroscedastic noise, without gaining a complexity penalty from losing long scale correlations.

Finally, in regime 3, FITC starts to behave more like a regular GP in terms of marginal likelihood, predictive performance and noise variance parameter σ_n^2. FITC's ability to use heteroscedastic noise is reduced as the approximate covariance matrix $Q_{\mathbf{ff}}$ is closer to the true covariance matrix $K_{\mathbf{ff}}$ when many (initial) inducing input are spread over the input space.

In the previous section we showed that after adding a new inducing input, a better minimum obtained without the extra inducing input could be recovered by clumping. So it is clear that the minimum that was found with fewer active inducing inputs still exists in the optimisation surface of many inducing inputs; the optimiser just does not find it.

Remark 7. *When running FITC with many inducing inputs its resemblance to the full GP solution relies on local optima. Solutions with the earlier issues are still preferred in terms of the objective function, but are not found.*

3.2.5 VFE is hindered by local optima

So far we have seen that the VFE objective function is a true lower bound on the marginal likelihood and does not share the same pathologies as FITC. Thus, when optimising, we really are interested in finding a global optimum. The VFE objective function is not completely trivial to optimise, and often tricks, such as initialising the inducing inputs with k-means and initially fixing the hyperparameters [Hensman et al., 2015b, 2013], are required to find a good optimum. Others have commented that VFE has the tendency to underfit [Lázaro-Gredilla and Figueiras-Vidal, 2009]. Here we investigate the underfitting claim and relate it to optimisation behaviour.

As this behaviour is not observable in our 1D dataset, we illustrate it on the pumadyn32nm dataset[2] (32 dimensions, 7168 training, 1024 test), see table 3.1 for the results of a representative run with random initial conditions and $M = 40$ inducing inputs.

Using a squared exponential ARD kernel with separate lengthscales for every dimension, a full GP on a subset of data identified four lengthscales as important to model the data while scaling the other 28 lengthscales to large values (in table 3.1 we plot the inverse lengthscales).

[2]Obtained from http://www.cs.toronto.edu/~delve/data/datasets.html.

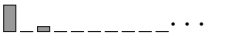
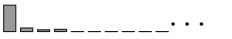
Method	**NLML**/N	σ_n	**inv. lengthscales**	**RMSE**
GP (SoD)	-0.099	0.196		0.209
FITC	-0.145	0.004		0.212
VFE	1.419	1		0.979
VFE (frozen)	0.151	0.278		0.276
VFE (init FITC)	-0.096	0.213		0.212

Table 3.1 Results for pumadyn32nm dataset. We show negative log marginal likelihood (NLML) divided by number of training points, the optimised noise variance σ_n^2, the ten most dominant inverse lengthscales and the RMSE on test data. Methods are full GP on 2048 training samples, FITC, VFE, VFE with initially frozen hyperparameters, VFE initialised with the solution obtained by FITC.

FITC was consistently able to identify the same four lengthscales and performed similarly compared to the full GP but scaled down the noise variance σ_n^2 to almost zero. The latter is consistent with our earlier observations of strong pinching in a regime with low-density data as is the case here due to the high dimensionality. VFE, on the other hand, was unable to identify these relevant lengthscales when jointly optimising the hyperparameters and inducing inputs, and only identified some of the them when initially freezing the hyperparameters. One might say that VFE "underfits" in this case. However, we can show that VFE still *recognises* a good solution: When we initialised VFE with the FITC solution it consistently obtained a good fit to the model with correctly identified lengthscales and a noise variance that was close to the full GP.

Remark 8. *VFE has a tendency to find under-fitting solutions. However, this is an optimisation issue. The bound correctly identifies good solutions.*

3.2.6 FITC can violate a marginal likelihood upper bound

In previous sections we saw that FITC deviates from the true GP solution mainly due to the objective function preferring to use the heteroskedastic correction term instead of noise. These solutions are assigned higher FITC marginal likelihoods compared to the maximum marginal likelihood of the full GP. Here, we analyse FITC's marginal likelihood in more detail by comparing it to the upper bound on the true marginal likelihood discussed in section 2.4.2. Our main aim is to assess whether the upper bound can identify when FITC over-estimates the marginal likelihood within a similar computational budget.

We start by training FITC models to a 100 point subset of the 1D dataset with increasing numbers of inducing points. We initialise the inducing points randomly from

the data, and perform 4 random restarts, choosing the model with the highest marginal likelihood. We then initialise all parameters jointly. We test whether the upper bound can identify any over-estimation in this setting within a similar computational budget as FITC has, so we limit the upper bound to have the same number of inducing points as FITC does. We test 3 different variations for choosing the inducing inputs for the upper bound: 1) the optimised values from the FITC model, and 2) optimised by minimising the upper bound itself. We optimise the inducing points of the upper bound by minimising it from the same initialisation.

Figure 3.6 shows a comparison between the upper bound and the marginal likelihood of a FITC model with increasing numbers of inducing points. We see that FITC over-estimates the marginal likelihood already with few inducing variables. The behaviour is the same as discussed in chapter 3, with very low noise levels being preferred. Once enough inducing points are added, the optimised upper bound becomes tight enough to detect that the FITC marginal likelihood is an over-estimate. Note that even though the over-estimation very significant compared to the full GP at its optimal parameters, FITC very strongly over-estimates the marginal likelihood for the hyperparameters it finds (i.e. with very low noise).

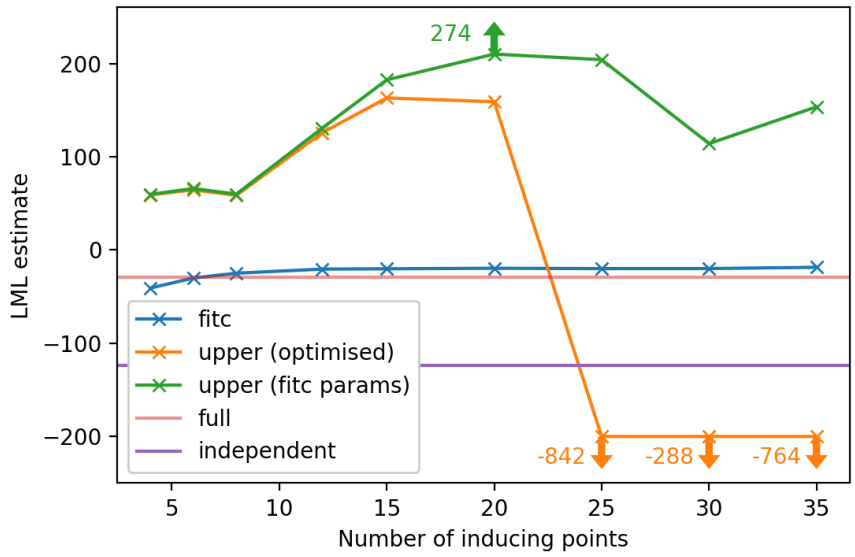


Fig. 3.6 Comparison of marginal likelihood upper bounds with FITC, full GP, and independent Gaussian (for reference) marginal likelihoods. Arrows with corresponding numbers indicate out of scale values.

We look at this data more closely in figure 3.7, where we show upper bounds with increasing numbers of inducing points for each FITC model in figure 3.6, to give an indication of when the bounds become useful. We see that when using a small number

of inducing points, FITC either correctly estimates the marginal likelihood (as for $M = 4$) the marginal likelihood, or has a small over-estimation ($M = 6$, $M = 8$), for its chosen hyperparameters. However, in this regime, the upper bound needs more inducing points than FITC to prove the over-estimation. When more inducing points are added, the upper bound can identify over-estimation, consistent with figure 3.6.

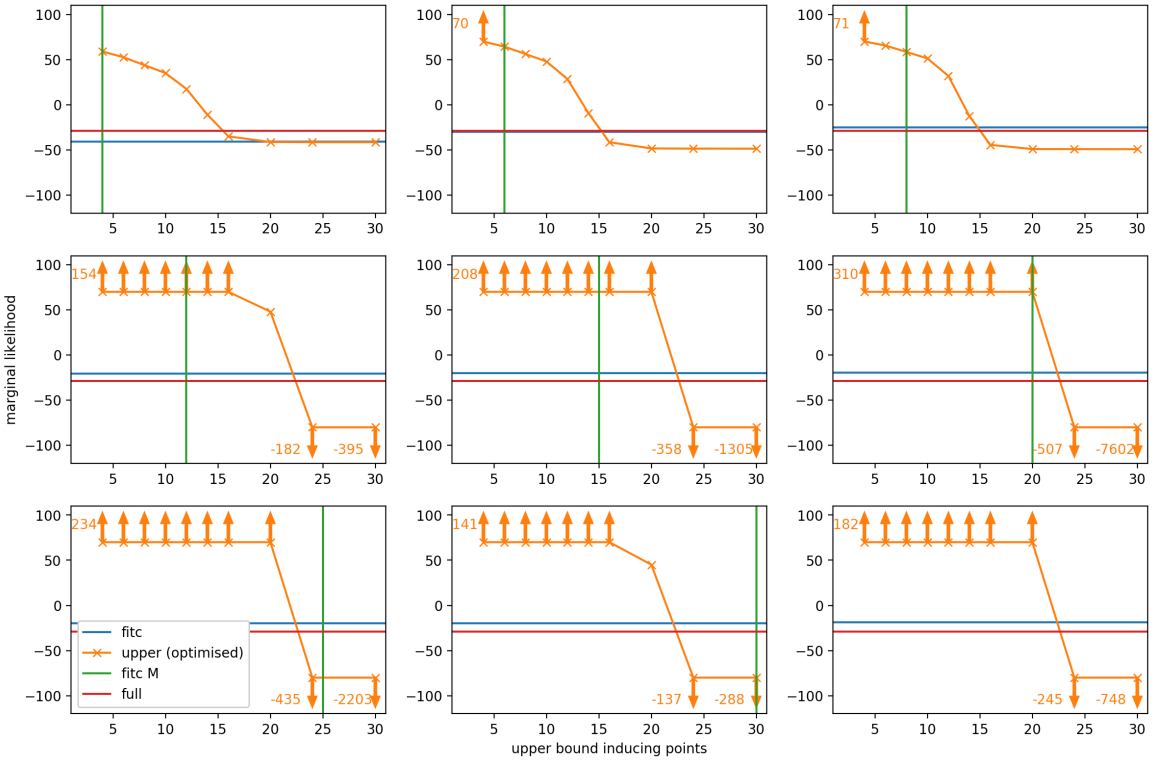


Fig. 3.7 Marginal likelihoods of FITC models with different numbers of inducing points (M), compared to upper bounds with increasing numbers of inducing points. Arrows indicate out of scale values, numbers indicate value.

3.2.7 Conclusion

In this work, we have thoroughly investigated and characterised the differences between FITC and VFE, both in terms of their objective function and their behaviour observed during practical optimisation.

We highlight several instances of undesirable behaviour in the FITC objective: the lack of guarantees of the FITC marginal likelihood and its consistent over-estimation when given enough inducing points, sometimes severe under-estimation of the noise variance parameter, wasting of modelling resources and not recovering the true posterior. The common practice of using the noise variance parameter as a diagnostic for good

model fitting is unreliable. In contrast, VFE is a true bound to the marginal likelihood of the full GP and behaves predictably: It correctly identifies good solutions, always improves with extra resources and recovers the true posterior when possible. In practice however, the pathologies of the FITC objective do not always show up, thanks to "good" local optima and (unintentional) early stopping. While VFE's objective recognises a good configuration, it is often more susceptible to local optima and harder to optimise than FITC.

The sparse upper bound to the marginal likelihood can be used to identify when FITC finds a solution that it over-estimates the marginal likelihood for, but only when enough inducing points are used. Unfortunately, over-estimation starts with a smaller number of inducing points

Which of these pathologies show up in practise depends on the dataset in question. However, based on the superior properties of the VFE objective function, we recommend using VFE, while paying attention to optimisation difficulties. These can be mitigated by careful initialisation, random restarts, other optimisation tricks and comparison to the FITC solution to guide VFE optimisation.

3.3 Parametric models, identical bounds

So far, we have seen that the a GP can often be well approximated by an inducing point approximation, where the inducing points are chosen through optimising an approximation to the full GP marginal likelihood. This approximation comes in the form of either an alternative, parametric model's marginal likelihood [Quiñonero-Candela and Rasmussen, 2005], or as a variational lower bound [Titsias, 2009b]. The main attraction of the variational lower bound is that it approximates the correct latent process with fewer computational resources, while retaining some of the desirable properties of the non-parametric posterior (section 2.4.2 and chapter 3).

It may then come as a disappointment (and perhaps a surprise) that the variational lower bound stemming from a non-parametric GP model, can be the same as from FITC, which can be seen as a parametric model. This was first noted by Frigola et al. [2014] in reference to the GPSSM, and highlighted as an unexplained disappointment. This same effect was noted for simple GP regression during personal communication with Yarin Gal. The observation that we can start with a parametric model, perform approximate inference, and end up with the *same* procedure as for a non-parametric model is worrying. After all, how could we expect *any* of the desirable properties of the

non-parametric posterior to be present in an approximation, if the same approximation is also obtained from a parametric model?

Conversely, if we perform inference in a model starting from FITC, perhaps expecting to replicate its heteroskedastic noise behaviour, to what degree should we expect this behaviour to be maintained if the approximation scheme we obtain is the same as for a non-heteroskedastic Gaussian process? Mixing variational inference and model approximations with the goal of maintaining some of the model properties has been attempted [Hoang et al., 2015, 2016], so an understanding of this correspondence should help to identify when model properties are maintained, and when they are not.

Here, we discuss the curious equivalence of variational bounds from FITC style models, and from the full GP. We characterise for what models there is a correspondence, and why properties of the non-parametric model *are* maintained in the approximation, while properties of FITC are not.

3.3.1 FITC and GP regression share a lower bound

We start with Titsias's [2009b] original finite-dimensional derivation for GP regression. We represent $N + M$ points on the Gaussian process, corresponding to the N data inputs X and outputs $\mathbf{f}$, and the M inducing inputs Z and outputs $\mathbf{u}$. We use the variational distribution $q(\mathbf{f}, \mathbf{u}) = q(\mathbf{u})p(\mathbf{f}|\mathbf{u})$, where $q(\mathbf{u})$ is a free-from distribution, and $p(\mathbf{f}|\mathbf{u})$ is the prior conditional under the Gaussian process.

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})\mathrm{d}\mathbf{f} = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \tag{3.4}$$

$$\geq \int q(\mathbf{u})p(\mathbf{f}|\mathbf{u}) \log \frac{\prod_i p(y_i|f_i)\cancel{p(\mathbf{f}|\mathbf{u})}p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})}q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \tag{3.5}$$

$$= \sum_{i=1}^{N} \int q(\mathbf{u})p(f_i|\mathbf{u}) \log p(y_i|f_i)\mathrm{d}f_i\mathrm{d}\mathbf{u} - \mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u}) \,] \tag{3.6}$$

$$:= \mathcal{L} \tag{3.7}$$

On inspection of the above bound, we see that it depends on the likelihood, prior over inducing points, and the *marginal* conditional prior $p(f_i|\mathbf{u})$. We can therefore construct models with the same lower bound, simply by choosing different covariances between the $\mathbf{f}$s but the same marginals. FITC fulfills these criteria, since it was

constructed to have the same variances along the diagonal:

$$p_{GP}(\mathbf{f}|\mathbf{u}) = \mathcal{N}\Big(\mathbf{f}; K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big)\,, \tag{3.8}$$

$$p_{FITC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}\Big(K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, \mathrm{diag}\Big[K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\Big]\Big)\,. \tag{3.9}$$

As a consequence, we have:

Remark 9. *The FITC marginal likelihood will always be greater than or equal to the variational free energy $\mathcal{L}$.*

Many other models relying on sparse variational GPs share this property, like the Bayesian GPLVM [Titsias and Lawrence, 2010], Deep Gaussian processes [Damianou and Lawrence, 2013], or GPSSMs [Frigola et al., 2014] where this relation was originally pointed out.

It is rather worrying that these two models share the same variational bound. Normally, it is assumed as fairly reasonable that a variational inference method should exhibit the properties of the model it was derived for, except for a few biases here and there. Now we have shown that the same variational bound can be derived from a model which behaves significantly differently in some situations (section 3.2). While the previous section should leave little doubt that the non-parametric GP is correctly approximated, the question of "why" this behaviour shows up is still unanswered. In the following, we aim to characterise when the bound behaves like either FITC or the full GP. We argue that it is a rather unnatural approximation to FITC and a natural approximation to the full GP.

3.3.2 $\mathcal{L}$ as an approximation to FITC

Given that Titsias's [2009b] approximation can also be seen as a variational approximation to FITC, we may be wonder whether the preference of FITC for under-estimated noise levels (chapter 3) can be exhibited by VFE as well, or if not, why? From the outset, this worry can be dismissed because of the trivial observation that for any setting of Z, $\mathcal{L}$ will be a lower bound to both the full GP and FITC marginal likelihoods. As a consequence, $\mathcal{L}$ simply cannot over-estimate the marginal likelihood of the full GP, which is the mechanism for FITC's behaviour. On the flip-side, the bound is also limited by the smaller of the FITC and full GP marginal likelihoods.

Remark 10. *If FITC does not have the capacity to model the data well (in a marginal likelihood sense), neither will the variational bound. Additionally, even if FITC has the capacity to over-fit, the variational bound will not, due to being bounded by the full GP.*

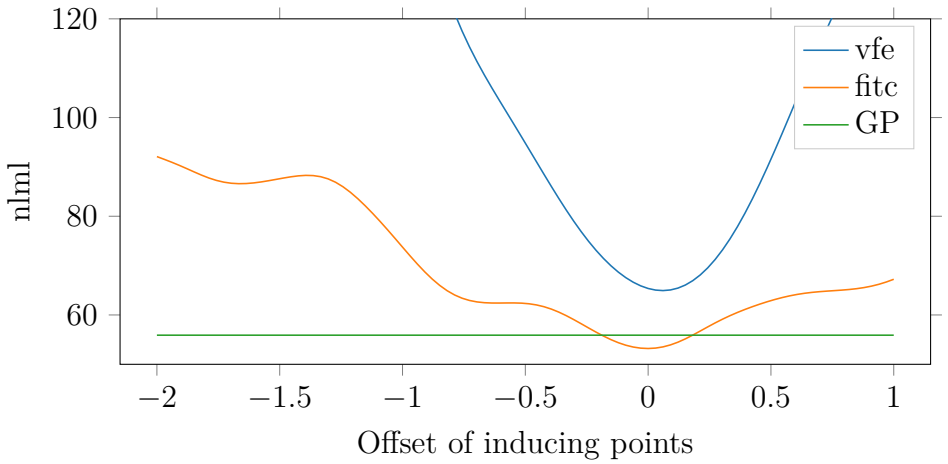


Fig. 3.8 The negative bound, FITC NLML and full GP NLML as a function of an offset added to *all* inducing points. The full GP remains unchanged, while FITCs NLML does change. The optimum for the bound is located at the point of least slack to the full GP. This is not the case for FITC, as the FITC NLML changes with the inducing point location as well. The underlying regression problem is the 1D example dataset figure 2.1.

We can also note another property from the outset, which will be the main explanation as to why $\mathcal{L}$ will not approximate FITC well: Z influences the marginal likelihood of FITC, while it leaves that of the full GP unchanged (figure 3.8). So for FITC, Z also takes the role of hyperparameters, while they are only variational parameters for the full GP. In fact, tying Z to be the same for both the prior and the approximate posterior is necessary to get a tractable variational bound. As a consequence:

Remark 11. *Optimising Z will make $\mathcal{L}$ closer to the full GP marginal likelihood, but not necessarily to the FITC marginal likelihood.*

We can investigate why $\mathcal{L}$ follows the full GP rather than FITC in more detail by taking a closer look at the true FITC and variational posteriors. In FITC, the true posterior of $\mathbf{u}$ is sufficient for making predictions on any new testing point. The posterior for new points can be found using the inducing conditional $p_{FITC}(\mathbf{f}|\mathbf{u})$. Given this observation, the approximate variational distribution seems sensible, and seems like it could represent the exact posterior. However, the optimal $q(\mathbf{u})$ for this bound is [Titsias, 2009b]:

$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}; \sigma^{-2} K_{\mathbf{uu}} \left(K_{\mathbf{uu}} + \sigma^2 K_{\mathbf{uf}} K_{\mathbf{fu}}\right)^{-1} K_{\mathbf{uf}} \mathbf{y}, K_{\mathbf{uu}} \left(K_{\mathbf{uu}} + \sigma^2 K_{\mathbf{uf}} K_{\mathbf{fu}}\right)^{-1} K_{\mathbf{uu}}\right). \tag{3.10}$$

Since this optimal distribution is a consequence of the bound, and *not* of the model, it remains unchanged regardless of whether the origin of the bound was the full GP or the FITC model. This variational posterior is different to the true FITC posterior, which can be obtained through Bayes' rule:

$$p(\mathbf{u}|\mathbf{y}) = \mathcal{N}\Big(\mathbf{u}; K_{\mathbf{uu}}\Big(K_{\mathbf{uu}} + K_{\mathbf{uf}}\Lambda^{-1}K_{\mathbf{fu}}\Big)^{-1}K_{\mathbf{uf}}\Lambda^{-1}\mathbf{y}, K_{\mathbf{uu}}\Big(K_{\mathbf{uu}} + K_{\mathbf{uf}}\Lambda^{-1}K_{\mathbf{fu}}\Big)^{-1}K_{\mathbf{uu}}\Big)\,. \tag{3.11}$$

This is puzzling, since no obvious constraints are placed on $q(\mathbf{u})$ during the derivation of the bound, hinting that it ought to be able to recover the true posterior, particularly when it is so easily represented.

An explanation for this behaviour can be found by considering the KL divergence that is minimised, rather than the bound:

$$p(\mathbf{y}) - \mathcal{L} = \mathrm{KL}[\, q(\mathbf{u})p_{FITC}(\mathbf{f}|\mathbf{u}, X, Z) \,\|\, p_{FITC}(\mathbf{u}, \mathbf{f}|\mathbf{y}, X, Z)\,]\,. \tag{3.12}$$

The bound also cares about matching the posterior at the training points $\mathbf{f}$, even though they are not strictly necessary for prediction at any new point. This is the root cause for the variational bound not approximating FITC. FITC as a model has a heteroscedastic noise component that considered to be part of the latent process f. As a consequence, any observation y_i will also inform how large the heteroscedastic noise component was in f_i – information which is *not* contained in $\mathbf{u}$.

The factorisation in the approximate posterior is not able to reduce the variance of the heteroscedastic component at the training locations, and so the true joint posterior $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$ can not be recovered. This, in turn, biases $q(\mathbf{u})$ away from the true FITC posterior, to minimise the joint KL, and explains why FITC is not well approximated. One notable exception is when an inducing point is placed on each input. In this case, FITC is equivalent to the full GP, and VFE correctly recovers the posterior [Snelson, 2007].

Remark 12. *The variational bound does* not *recover the FITC posterior, due to the heteroscedastic noise component being considered part of the process. As a consequence, the variational posterior can never capture FITC's, except when inducing points are placed at all inputs, where FITC, VFE and the full GP are all equivalent.*

3.3.3 Conclusion

Based on the observations above, it seems as though the variational bound can be seen as a severely hampered approximation to FITC. It will not recover the FITC posterior, will always return an under-estimated marginal likelihood, and there are no free variational parameters to tighten the bound. Optimising Z is not guaranteed to minimise the KL to the FITC model. In contrast, optimising Z does minimise the KL to the full GP. The bound will also recover the full GP, given enough resources. So from all perspectives VFE is a very poor approximation to FITC, but a good one to the full GP.

It is interesting that hampering FITC in this way results back in an approximation to the non-parametric full GP. We offer no deep insight into why this is the case. The only intuition is based on FITC having the tendency to over-fit when given enough inducing inputs, in contrast to the full GP, and that this capacity is removed by the constraint of the variational approximation. This constraint is apparently just right to give back properties of the non-parametric model.

Overall, we hope that we have explained why FITC and VFE share a variational lower bound, and why VFE does behave as one would want an approximation to.

Chapter 4

Inter-domain basis functions for flexible variational posteriors

In the previous sections we discussed how to approximate full Gaussian process posteriors using cheaper Gaussian processes that were only conditioned on M points. In chapter 3 we discussed how the variational approximation is particularly appealing since it can get very close to the posterior and optimal hyperparameters for the true GP, if it is given enough inducing points. Unfortunately, the cost for adding inducing points is cubic so in practice we are still limited to approximations with only a few thousand inducing variables. For datasets that consist of millions of data points, how sure are we that this is really enough to get close to the true Gaussian process posterior?

In order to get the most out of a sparse approximation, we ought to get the most out of each individual inducing point, to avoid the expense of using too many. Optimising the inducing points is a first very helpful step. A further improvement was suggested by the remark in Titsias [2009a] that "any linear functional involving the GP function $f(x)$ can be an inducing variable", since any linear transformation of $f(\cdot)$ would also have a Gaussian distribution. This was subsequently taken up by Lázaro-Gredilla and Figueiras-Vidal [2009], who used smoothing and wavelet-like integral transforms to define the inducing variables, and selected them using the FITC objective (Snelson and Ghahramani [2006], chapter 3). While Lázaro-Gredilla and Figueiras-Vidal [2009] dismissed using the VFE objective due to the perception of its tendency to under-fit (addressed in section 3.2.5), it was later used by Alvarez et al. [2010] for implementing multi-output kernels.

Here we revisit the inter-domain inducing variables and propose a method to squeeze even more information out of each variable. While Lázaro-Gredilla and Figueiras-Vidal [2009] propose fixed inter-domain transformations, we propose to parameterise the

transformation itself using basis functions. This allows us to add a cheaper type of basis function to the approximate posterior, thereby increasing its flexibility *without* increasing the size expensive matrix operations. We investigate how this variational posterior behaves.

In the next sections we will describe and analyse inter-domain approximations, particularly from the point of view of variational inference. We will then justify our proposed method, and present experimental results showing that inter-domain basis functions can indeed be used to sparsify models. While practical benefits to training speed are small, the method can be used for compressing Gaussian process models in terms of storage size.

4.1 Related work

Inter-domain inducing variables were first proposed together with the variational approximation framework by Titsias [2009a]. Later, Matthews [2016] showed that variational posteriors using inter-domain inducing variables also minimise the KL divergence between the latent stochastic processes, if the relationship between the inter-domain variables and the latent GP is deterministic.

Walder et al. [2008] first introduced the idea of using basis functions in the approximation that were different to the kernel. Their specific approach did not correspond to a well-formed Gaussian process, as it could exhibit negative predictive variances, which was fixed in Lázaro-Gredilla and Figueiras-Vidal [2009], where they were correctly combined with FITC objective for training. Additionally, they have seen use in situations where placing inducing points in the usual input domain would lead to a useless approximation. Alvarez et al. [2010] and Tobar et al. [2015] both use white noise to drive a filter, the output of which is then observed. In these cases, individual function values have no correlation with the rest of the process, and are also not useful for predicting the data. Instead, the lower-frequency behaviour of the process is of importance, and can be captured by inter-domain inducing variables.

Hensman et al. [2018] propose a method most in the spirit of this work, where Fourier-like inter-domain features are used. Very impressive speed-ups were reported, but the method suffers from a curse of dimensionality, so scaling this method is constrained to using additive models in high dimensions.

4.2 Inter-domain inducing variables

As mentioned, the main idea is to use general linear functionals of $f(\cdot)$ as inducing variables, rather than $f(Z)$. This can be a good idea, since it captures more global properties about the function than a single point does. Later we will also see that it gives more flexibility to the mean of the posterior process. An inter-domain inducing variable is constructed by taking an integral projection of the GP $f(\cdot)$ with respect to some function $v(\cdot)$, parameterised by $\mathbf{z}$:

$$u(\mathbf{z}) = \int f(\mathbf{x})v(\mathbf{x};\mathbf{z})\mathrm{d}\mathbf{x}\,. \tag{4.1}$$

Since $f(\cdot)$ is a random variable, $u(\mathbf{z})$ will be as well, and since the projection is linear, it will also be Gaussian. We can find its mean, covariance and cross-covariance by substituting in its definition, and taking expectations over $f(\cdot)$. As usual, we take $f \sim \mathcal{GP}(0, k(\mathbf{x},\mathbf{x}'))$.

$$\mathbb{E}_f[u(\mathbf{z})] = \mathbb{E}_f\Big[\int f(\mathbf{x})v(\mathbf{x};\mathbf{z})\mathrm{d}\mathbf{x}\Big] = \int v(\mathbf{x};\mathbf{z})\mathbb{E}_f[f(\mathbf{x})]\mathrm{d}\mathbf{x} = 0 \tag{4.2}$$

$$\begin{aligned}\mathrm{Cov}_f[u(\mathbf{z}), f(\mathbf{x})] &= \mathbb{E}_f\Big[\int f(\mathbf{x}')v(\mathbf{x}';\mathbf{z})\mathrm{d}\mathbf{x}' f(\mathbf{x})\Big] = \int v(\mathbf{x}';\mathbf{z})\mathbb{E}_f[f(\mathbf{x}')f(\mathbf{x})]\mathrm{d}\mathbf{x}' \\ &= \int v(\mathbf{x};\mathbf{z}')k(\mathbf{x},\mathbf{x}')\mathrm{d}\mathbf{x}' \end{aligned} \tag{4.3}$$

$$\begin{aligned}\mathrm{Cov}_f[u(\mathbf{z}), u(\mathbf{z}')] &= \mathbb{E}_f\Big[\Big(\int f(\mathbf{x})v(\mathbf{x};\mathbf{z})\mathrm{d}\mathbf{x}\Big)\Big(\int f(\mathbf{x}')v(\mathbf{x}';\mathbf{z})\mathrm{d}\mathbf{x}'\Big)\Big] \\ &= \iint v(\mathbf{x},\mathbf{z})v(\mathbf{x}',\mathbf{z}')k(\mathbf{x},\mathbf{x}')\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}' \end{aligned} \tag{4.4}$$

Using these covariances, we can calculate the covariance matrices $K_{\mathbf{f}^*\mathbf{u}}$, $K_{\mathbf{uu}}$ and $K_{\mathbf{uf}}$, needed for representing the sparse approximation. It is worth emphasizing how easy it is to use inter-domain variables with *any* of the inducing variable techniques described in section 2.3. These approximations only depend on the (cross-) covariances between $\mathbf{u}$ and $\mathbf{f}$, and so the expressions above are a simple drop-in replacement.

The practical usability of inter-domain methods depends on whether the integrals to find these covariances can be computed analytically. Lázaro-Gredilla and Figueiras-Vidal [2009] point out that one can integrate Gaussians and sinusoids against other Gaussians, and use this to propose three possible projections: multi-scale (MSIF), frequency (FIF), and time-frequency (TFIF). In the following statement of the projections we denote the parameters of the inducing input $\mathbf{z}$ to contain whichever are present of

$\boldsymbol{\mu}$, Δ, $\boldsymbol{\omega}$ and ω_0. Δ is usually taken to be $\text{diag}\begin{bmatrix}\delta_1^2 & \cdots & \delta_D^2\end{bmatrix}$.

$$v_{MS}(\mathbf{x};\mathbf{z}) = |2\pi\Delta|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal\Delta^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{4.5}$$

$$v_F(\mathbf{x};\mathbf{z}) = |2\pi\Delta|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{x}^\intercal\Delta^{-1}\mathbf{x}\right)\cos(\omega_0+\boldsymbol{\omega}^\intercal\mathbf{x}) \tag{4.6}$$

$$v_{TF}(\mathbf{x};\mathbf{z}) = |2\pi\Delta|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal\Delta^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\cos(\omega_0+\boldsymbol{\omega}^\intercal(\mathbf{x}-\boldsymbol{\mu})) \tag{4.7}$$

We reproduce the cross-covariances here as they will illuminate the methods later. We also note that the expression of $k(\mathbf{z},\mathbf{z}')$ for the Time-Frequency inducing features was not correctly reported in either Lázaro-Gredilla and Figueiras-Vidal [2009] or the later Lázaro-Gredilla [2010]. We derive the correct covariance in appendix A.1.

$$k_{MS\mathbf{fu}}(\mathbf{x},\mathbf{z}) = \prod_{d=1}^{D}\left(\frac{l_d^2}{l_d^2+\delta_d^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal(\Delta+\Lambda)^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{4.8}$$

$$k_{F\mathbf{fu}}(\mathbf{x},\mathbf{z}) = \prod_{d=1}^{D}\left(\frac{l_d^2}{l_d^2+\delta_d^2}\right)^{\frac{1}{2}} \exp\left(\sum_{d=1}^{D}\frac{x_d^2+\delta_d^2 l_d^2\omega_d^2}{2(\delta_d^2+l_d^2)}\right)\cos\left(\omega_0+\sum_{d=1}^{D}\frac{\delta_d^2\omega_d x_d}{2(\delta_d^2+l_d^2)}\right) \tag{4.9}$$

4.3 Inter-domain posteriors

To understand what benefit we may get from inter-domain inducing variables, we first analyse the structure of the posterior, and how it relates to the bound we're optimising. In section 2.4.1 we discussed the structure and properties of the approximate Gaussian process when using inducing points. They could be split into components $g(\cdot)$ and $h(\cdot)$, having the forms:

$$f \sim \mathcal{GP}\left(k_{*\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}, K_{\mathbf{f}^*\mathbf{f}^*} - K_{\mathbf{f}^*\mathbf{u}}K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}}-\Sigma)K_{\mathbf{uu}}^{-1}K_{\mathbf{f}^*\mathbf{u}})\right), \tag{4.10}$$

$$f(\cdot) = g(\cdot) + h(\cdot), \tag{4.11}$$

$$g(\cdot) \sim \mathcal{GP}\left(k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}; k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\Sigma K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\right), \tag{4.12}$$

$$h(\cdot) \sim \mathcal{GP}\left(0, k(\cdot,\cdot) - k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\right). \tag{4.13}$$

We saw in figure 2.4 how an inducing point added capacity to the $g(\cdot)$ which could adapt to data, and locally reduced the variance in $h(\cdot)$, which represented the variance that could not be explained by the inducing points. By changing the covariance relationships between $\mathbf{u}$ and $\mathbf{f}$, we change the capacities of $g(\cdot)$ and $h(\cdot)$. Foremost, we notice that different basis functions are used for the mean in $g(\cdot)$. Instead of being the usual kernel function, one of the cross-covariance functions (equation 4.8) is used. For

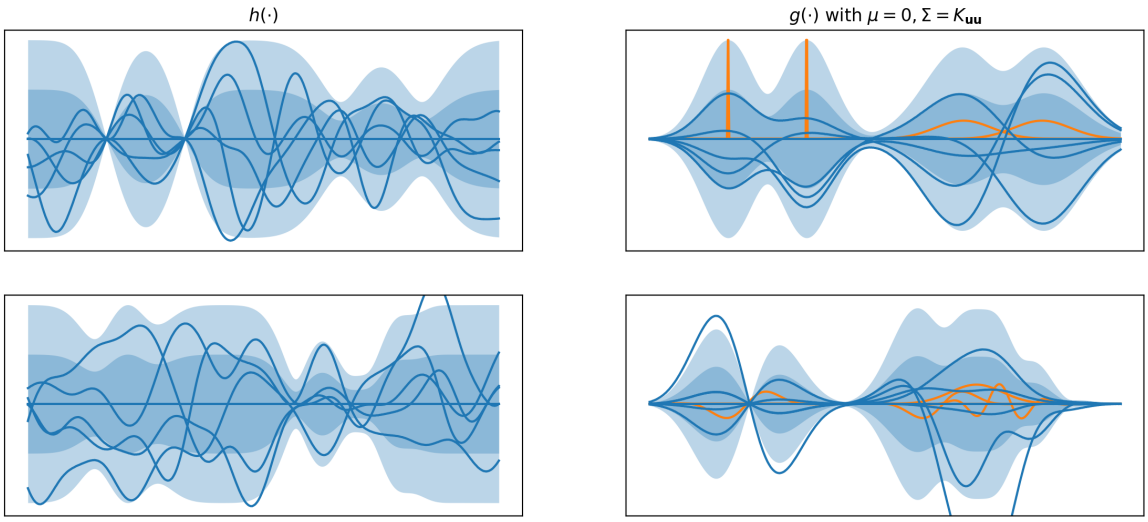


Fig. 4.1 The exact GP prior split into its components $h(\cdot)$ and $g(\cdot)$ as in figure 2.4. Top: Multi-scale inducing features, with ones tending to inducing points in the left for comparison. Bottom: Frequency inducing features.

the multi-scale case, this means that longer lengthscales are used. These lengthscales can be optimised, which can be useful to fit some regions in the posterior where there may not be a lot of variation in the mean. The frequency inducing feature also seems attractive, since it gives many local extrema – something which could only be achieved with several inducing points. Another effect is that the variance reduction in $h(\cdot)$ gets spread out over a larger range, at the expense of it reducing the variance at a point to zero. We see this practically in figure 4.1 for the multi-scale and time-frequency inducing features. The multi-scale features are wider, while the frequency features allow for more extrema.

Essentially, the inter-domain features allow a trade-off between the complexity of the mean of the posterior process against its covariance structure and how much it can be reduced. Initially, this seems like a very sensible idea. In noisy regression, it seems unreasonable to expect the data be able to reduce the uncertainty in the posterior process to *exactly* zero, so why do we need a class of approximate posteriors that spends computational resources to represent this? Lázaro-Gredilla and Figueiras-Vidal [2009] showed that FITC models could be made significantly sparser with the more complicated inducing features from equation 4.8.

4.4 Basis functions for inter-domain projections

Rather than committing to a fixed form of the inter-domain variable, defined by a particular form of the projection $v(\cdot, \cdot)$, we propose to parameterise it directly using basis functions $\tilde{v}(\cdot, \cdot)$, and let the objective function decide what the best projection is:

$$v(\mathbf{x}; \mathbf{z}) = \sum_{c=1}^{C} w_c \tilde{v}(\mathbf{x}, \mathbf{z}^{(c)}), \qquad \mathbf{z} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(C)}, \mathbf{w}\}. \tag{4.14}$$

The parameters of the overall projection are the parameters for each basis function $\mathbf{z}^{(c)}$ (e.g. its location and scale for MSIF) and the vector of weights $\mathbf{w} = [w_1 \cdots w_c]^\mathsf{T}$. We can choose any function that is integrable against the kernel as the basis function v_{bf}. For the squared exponential kernel, this could be any of the equations 4.5 to 4.7. Since summing is a linear operation, the cross-covariance with the observations $K_{\mathbf{fu}}$ and the covariance of the inducing points $K_{\mathbf{uu}}$ simply become sums of the covariance $\tilde{k}$ corresponding to the projection basis function $\tilde{v}$.

$$k_{\mathbf{fu}}(\mathbf{z}, \mathbf{x}) = \mathbb{E}_f \left[\int f(\mathbf{x}) \sum_{c=1}^{C} w_c \tilde{v}\left(\mathbf{x}; \mathbf{z}^{(c)}\right) \mathrm{d}\mathbf{x} \right] = \sum_{c=1}^{C} w_c \tilde{k}_{\mathbf{fu}}\left(\mathbf{x}, \mathbf{z}^{(c)}\right) \tag{4.15}$$

$$\begin{aligned} k_{\mathbf{uu}}(\mathbf{z}, \mathbf{z}') &= \mathbb{E}_f \left[\left(\int f(\mathbf{x}) \sum_{c=1}^{C} w_c \tilde{v}\left(\mathbf{x}; \mathbf{z}^{(c)}\right) \mathrm{d}\mathbf{x} \right) \left(\int f(\mathbf{x}') \sum_{c'=1}^{C} w_{c'} \tilde{v}\left(\mathbf{x}'; \mathbf{z}'^{(c')}\right) \mathrm{d}\mathbf{x}' \right) \right] \\ &= \sum_{cc'} w_c w_{c'} \tilde{k}_{\mathbf{uu}}\left(\mathbf{z}^{(c)}, \mathbf{z}'^{(c)}\right) \end{aligned} \tag{4.16}$$

If we again consider that $k_{\mathbf{fu}}$ determines the basis functions of our posterior mean, we see that the summed inducing variables gives us $M \times C$ basis functions, rather than the usual M:

$$\mu_{posterior}(\cdot) = k_{\cdot\mathbf{u}}\boldsymbol{\alpha} = \sum_{m=1}^{M} \alpha_m k_{\mathbf{fu}}(\cdot, \mathbf{z}_m) = \sum_{m=1}^{M} \sum_{c=1}^{C} \alpha_m w_c \tilde{k}_{\mathbf{fu}}\left(\cdot, \mathbf{z}_m^{(c)}\right). \tag{4.17}$$

Delta inter-domain features

Here, we also would like to point out that we could simply take $\tilde{v}(\mathbf{x}, \mathbf{z}) = \delta(\mathbf{x} - \mathbf{z})$ – the Dirac delta function. By itself this is not that useful, since the resulting inter-domain covariances are just the same as for the inducing point case. However, within the basis function projections it does usefully increase the flexibility of the posterior. This also widens the range of kernels that inter-domain approximations can be applied to, over the squared exponential kernel discussed in Lázaro-Gredilla and Figueiras-Vidal [2009],

and is simple to implement. Figure 4.2 shows the structure of approximate posterior as earlier.

In the following experiments, we will focus on the "sum of deltas" inducing features, because of the generality in which they apply, but also because it allows a direct comparison of the trade-off between using normal inducing points, and adding more basis functions in a single inducing variable.

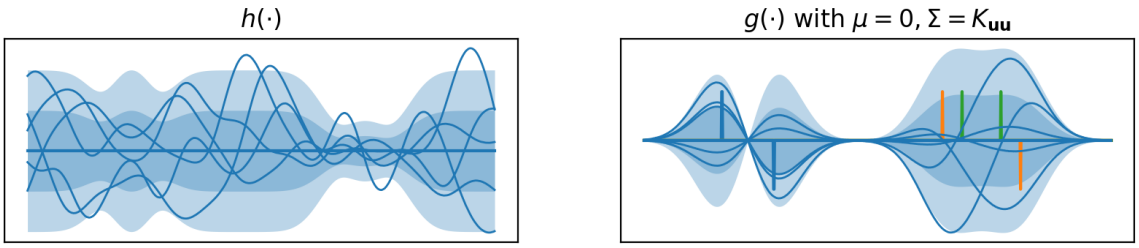


Fig. 4.2 Sum of delta functions inter-domain inducing features. Left: $h(\cdot)$. Right: $g(\cdot)$.

4.4.1 Computational complexity

Training using methods that require batch optimisation, like VFE or FITC, both require computing the entire bound and its gradients before updating the variational- and hyperparameters. The expensive operations for both these objective functions is computing the Cholesky decomposition of $K_{\mathbf{uu}} = LL^\intercal$, and calculating $L^{-1}K_{\mathbf{uf}}$ using a matrix triangular solve. These operations cost $\mathcal{O}(M^3)$ and $\mathcal{O}(NM^2)$ respectively. Using inter-domain basis functions does not increase the size of the matrices, leaving these costs unchanged. Evaluating the kernels does cost more, however. Evaluating $K_{\mathbf{fu}}$ (equation 4.15) costs $\mathcal{O}(NMC)$, while $K_{\mathbf{uu}}$ (equation 4.16) costs $\mathcal{O}(M^2C^2)$. The number of variational parameters scales linearly with C. It seems that in cases where the matrix operations are the bottleneck, inter-domain basis functions can give extra capacity at the roughly linear cost it takes to calculate the extra derivatives.

Hensman et al. [2013] introduced stochastic optimisation to GPs by modifying Titsias's [2009b] variational lower bound so an unbiased estimate can be found using a random sample of the training data. Computationally, the main difference is that a minibatch of size $\tilde{N} << N$ is used, which greatly reduces the cost of the matrix triangular solve operation to $\mathcal{O}\left(\tilde{N}M^2\right)$, leaving the bottleneck to be the $\mathcal{O}(M^3)$ Cholesky decomposition. In this case, C could be increased without making the bottleneck worse, until the kernel evaluations become the bottleneck.

Theoretically, inter-domain basis functions can also be used to reduce memory use. The matrices $K_{\mathbf{fu}}$ and $K_{\mathbf{uu}}$ need to be computed and stored. These can be calculated by accumulating each term in the sum block by block. Specifically, denoting the parameters of the cth basis function for all M inducing points by $Z^{(c)}$, we can write:

$$K_{\mathbf{fu}} = \sum_{c=1}^{C} w_c \tilde{k}\Big(X, Z^{(c)}\Big) = \sum_{c=1}^{C} w_c \tilde{K}_{\mathbf{fu}}^{(c)}\,, \tag{4.18}$$

$$K_{\mathbf{uu}} = \sum_{c=1}^{C}\sum_{c'=1}^{C} \tilde{k}\Big(Z^{(c)}, Z^{(c')}\Big) = \sum_{cc'} w_c w_{c'} \tilde{K}_{\mathbf{uu}}^{(c,c')}\,. \tag{4.19}$$

Implementation These matrices can be calculated using NM and M^2 numbers of memory space respectively, without growing with C, by accumulating the results directly. However, computing the gradients using reverse-mode automatic differentiation usually requires all the intermediate results $\tilde{K}_{\mathbf{fu}}^{(c)}$ and $\tilde{K}_{\mathbf{uu}}^{(c,c')}$ to be stored at a memory cost of $\mathcal{O}(NMC)$ and $\mathcal{O}(M^2C^2)$. In practice, this is a large problem, especially on GPUs where memory is limited to – at a push – 12GB.

While the $\mathcal{O}(NMC)$ cost is unavoidable, the more painful $\mathcal{O}(M^2C^2)$ is. Either, $K_{\mathbf{uu}}$ could be re-computed during the backwards accumulation at the cost of extra computation, or reverse-mode only autodiff should be avoided. Instead it is possible to compute the gradient as $\frac{\partial \mathcal{L}}{\partial K_{\mathbf{uu}}}\frac{\partial K_{\mathbf{uu}}}{\partial Z}$. Both intermediate gradients can be stored at a cost of $\mathcal{O}(M^2C)$, thanks to a sparsity pattern that the derivative of $K_{\mathbf{uu}}$ with a single inducing point has. The disadvantage of these suggestions is that they require some changes in the internal workings of the autodiff software packages. In TensorFlow [Abadi et al., 2015], which is used here for the implementation, this is (at the moment of writing) particularly hard. For this reason, we go ahead with the regular reverse-mode autodiff in our experiments.

4.4.2 A motivating example

In order to gain more intuition into where this method can present a benefit, we return to the 1 dimensional Snelson dataset. In the past, much attention has been paid to the perceived property of the variational bound to spread the inducing points evenly around the data [Titsias, 2009b] (often in contrast to FITC, see chapter 3). However, in sparse scenarios, we find that sometimes it is possible to obtain a much better posterior using two *almost* identical inducing variables. We see an example of this in figure 4.3. The variational bound uses two almost identical inducing variables to obtain quite a satisfactory solution. However, if the location of the inducing variable is modified

even in the slightest bit, the approximation deteriorates drastically, which is reflected correctly by the variational bound.

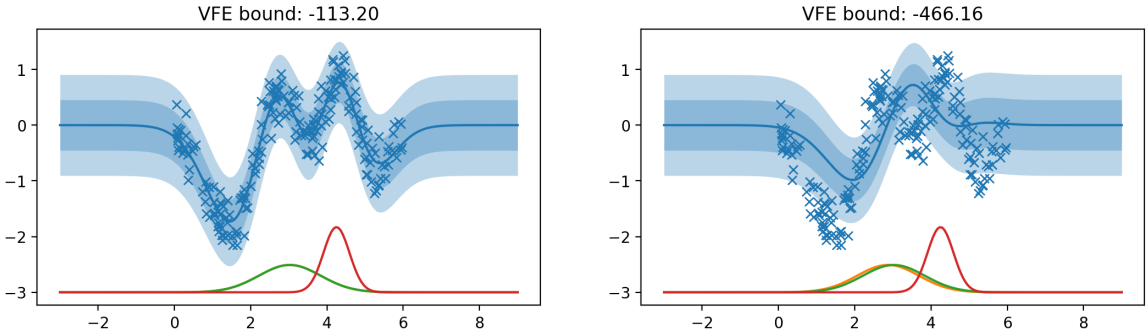


Fig. 4.3 Very sparse solution to the Snelson regression task. Multi-scale inducing features are used, and shown below the regression (for clarity, the projection value is zero at the start and finish). Left: The optimal solution. Note that 3 inducing features are used, but two are so close they show up as a single one. Right: The location of one of the inducing features is perturbed, causing the solution to become poor. The third inducing feature just becomes visible.

This solution is correct, and consistently found by the optimiser, and also surprising, since the intuition usually is that inducing points lose their representational capacity as they come close together. Figure 4.4 shows the effect that moving the inducing point around has on the bound. While it is the case that *exactly* repeating an inducing point effectively removes it and returns the bound to a low value, there is a very significant peak slightly to the right of it.

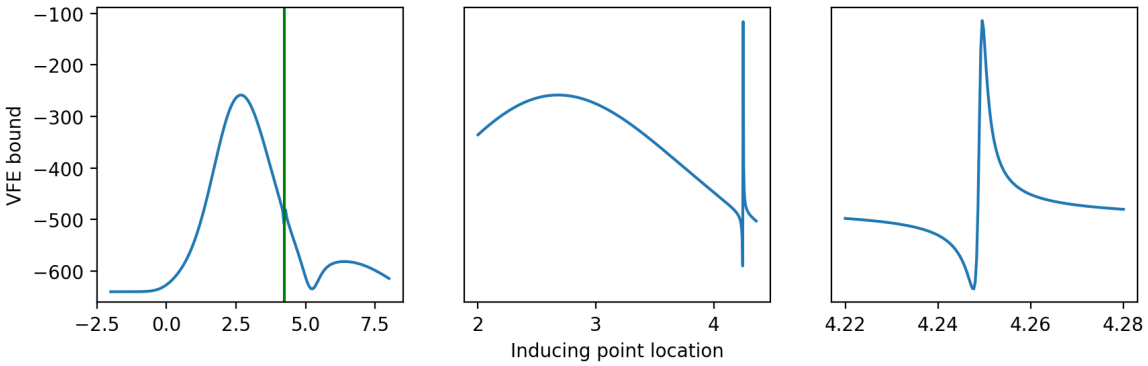


Fig. 4.4 Effect of the position of an inducing variable on the approximation quality, as measured by the variational lower bound. At first sight (left), the maximum appears to be around 2.6. However, the true peak only becomes visible after some zoom, and is around 4.2.

At first, this effect was thought to be caused by numerical instability, since having two inducing inputs that similar very strongly correlates their inducing outputs. Indeed, adding jitter removes this effect, as it limits the strongest possible correlation in $K_{\mathbf{uu}}$. However, in this case, the effect is actually favourable, as it allows a good posterior to be found. Looking at the resulting $q(\mathbf{u})$ for the similar variables, we find that their correlation is 0.99999979, while the means of the effective basis function weights ($\boldsymbol{\alpha} = K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}$) were $1.8662 \cdot 10^5$ and $-1.8674 \cdot 10^5$. Essentially, the variational approximation was using the difference between the two basis functions as a single basis function. Using the summed inducing projections, we can merge the effect of these two inducing points into one (figure 4.5). We see a slight penalty in the bound, but the approximate posterior is essentially unchanged, the mean in particular.

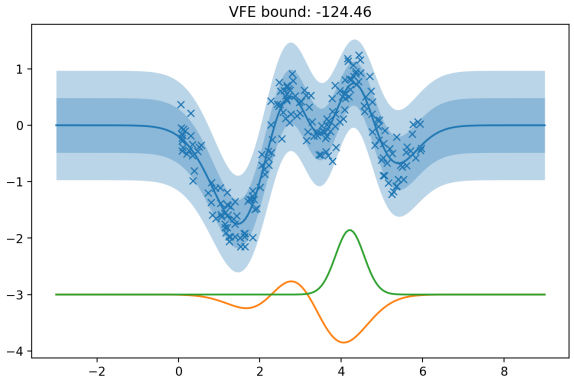


Fig. 4.5 Solution to the regression problem using 2 inducing features. One inducing feature is the sum of two multi-scale features, scaled by the weights set in the approximate GP solution.

We now move on from this toy dataset and experiment with the method on more realistic examples.

4.5 Experiments

Over the next several sections we will investigate the utility of these new basis functions. We will assess their behaviour on several UCI regression tasks. We are most interested in assessing what level of sparsity can be obtained in the approximate posterior by pushing the inducing variables to their limits, while still approximating the true posterior well. Given this objective, and in light of the results that FITC often does not approximate the model of interest (chapter 3), we will mainly focus on using the

variational free energy (VFE) bound to choose the approximation. We use the squared exponential kernel for all experiments.

For most of the following, we will optimise the hyperparameters jointly with the variational parameters, as is usual in sparse GP regression, unless stated otherwise. We will also use the "sum of delta" inducing features, proposed earlier, as this allows a direct comparison between adding basis functions using the most common inducing points, and adding basis functions in the inter-domain transform. We initialise the locations of the deltas by the randomly selecting the desired number of points from the dataset. Initially, we believed this may lead to poor solutions, as the points could lie far away from each other, leading to a single inducing variable (and therefore a single basis function) needing to explain distant parts of the GP with little correlation between them. To avoid this, we also tried performing a crude hierarchical clustering, first splitting the inputs into M clusters, and then clustering the points in each cluster again into C parts, using the latter centres as an initialisation. These procedures were not found to have significant impact on the final solution.

We will evaluate the method based on the quality of the approximation of the marginal likelihood, and the RMSE and NLPP it provides. We will also discuss hyperparameter selection for further insight into why the model is behaving as it does.

4.5.1 Sparsity of the resulting models

Firstly, we examine the extra sparsity that inter-domain basis functions can give us, similar to the experiments presented in Lázaro-Gredilla and Figueiras-Vidal [2009]. While this isn't necessarily a fair discussion in a practical sense since the inter-domain variables have many more free parameters to optimise, it helps give a sense of how sparse the posterior *can* be, given powerful enough inducing variables. We will first review two UCI datasets in their standard form. However, both datasets require an impractically large number of inducing points to actually recover the true posterior. Based on this observation, we add noise to both datasets, making it easier to recover the true posterior with a smaller M in order to investigate whether inter-domain basis functions help to recover the true posterior earlier.

Kin40k

We start with the "kin40k" dataset, with a 36k point training split. This dataset is particularly interesting since other GP approximations have not been able to get close to the full GP in terms of performance [Snelson and Ghahramani, 2006; Lázaro-

Gredilla and Figueiras-Vidal, 2009]. Figure 4.6 shows the behaviour of the sum of deltas projection, with changing numbers of inducing variables M and inter-domain bases C.

We see very large gains in the very sparse regime, especially for the RMSE. A model with 10 inducing variables and 20 inter-domain bases (200 bases in total), would seem to out-perform a model with 150 regular inducing variables. Gains for the NLPP are similarly large in the very sparse regime. However, the benefit of adding extra inducing variables is much smaller when the inter-domain projection is already complex. As more inducing variables are added, the benefit of inter-domain bases decreases, eventually leading to no additional benefit, even if the performance is still sub-optimal. The variational free energy reflects this as well.

When investigating the inferred noise level we see exactly the same pattern of improvement as in the other plots. In chapter 3 we argued that over-estimating the noise is a failure mode of VFE in very sparse models. Later, we will investigate whether hyperparameter estimation is the limiting factor for the improvement with added inter-domain bases.

Finally, we observe that sometimes the variational objective function prefers a posterior that performs worse under our chosen metrics. For example, $M = 200, C = 1$ has a better objective than $M = 50, C = 8$, despite having a worse RMSE.

Parkinsons

We see similar, but more encouraging, results for the "Parkinsons" dataset. In the very sparse regime we see very large improvements in RMSE. Using 5 inter-domain basis functions even achieves near-optimal RMSE at 100 basis functions, rather than > 200 for normal inducing points. Additionally, a considerable improvement in NLPP is seen even as the number of inducing points is increased, in contrast to kin40k. In this case, the benefit of inter-domain basis functions is clear all the way up to the regime where optimal performance is achieved.

Noisy kin40k

We now manually add Gaussian noise with a standard deviation of 0.4 to the kin40k dataset[1] to make it easier to capture with fewer inducing points. We do this to simulate a regime where the task *can* actually be solved to an optimal degree within the computational budget.

[1]Due to the noise, the maximum attainable performance will be worse than earlier. The lower bounds to the RMSE and NLPP will be 0.4 and 0.5 respectively.

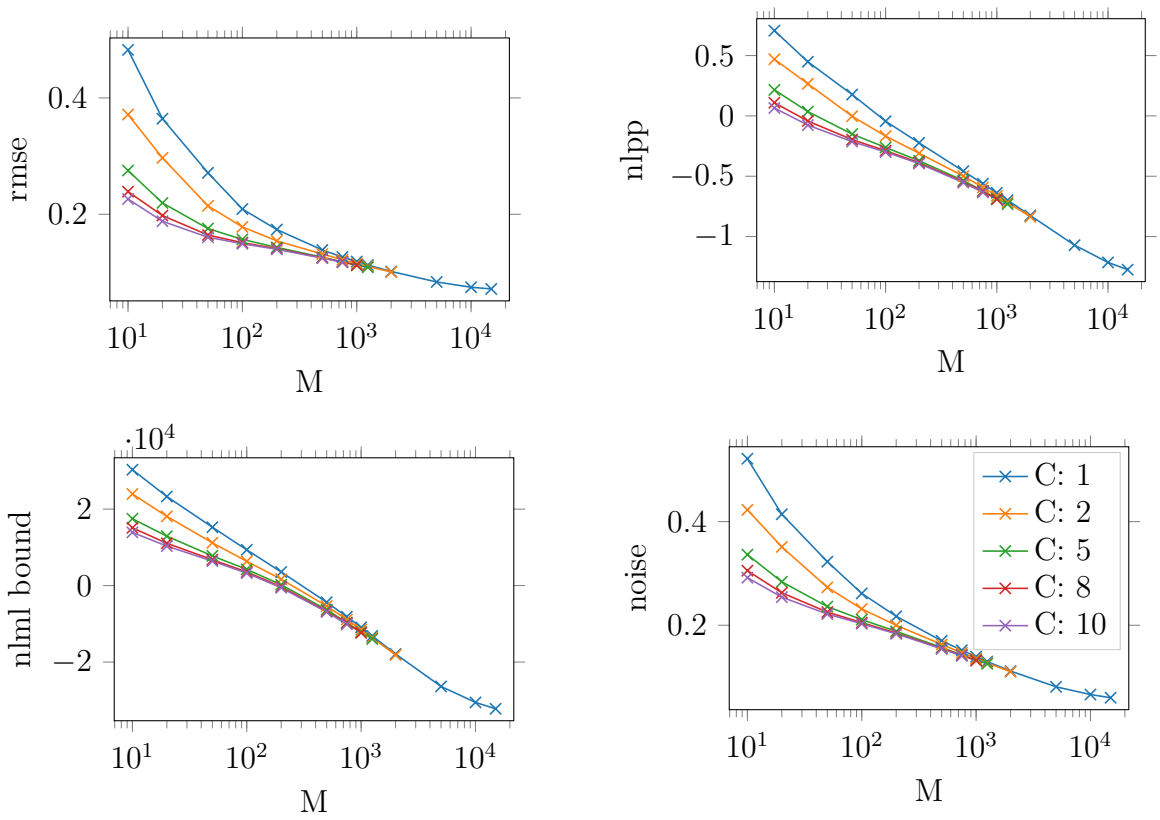


Fig. 4.6 Results for 36k split of kin40k, showing performance in RMSE and NLPP (top left and right) and the negative bound on the log marginal likelihood (NLML) and estimated noise hyperparameter (bottom left and right).

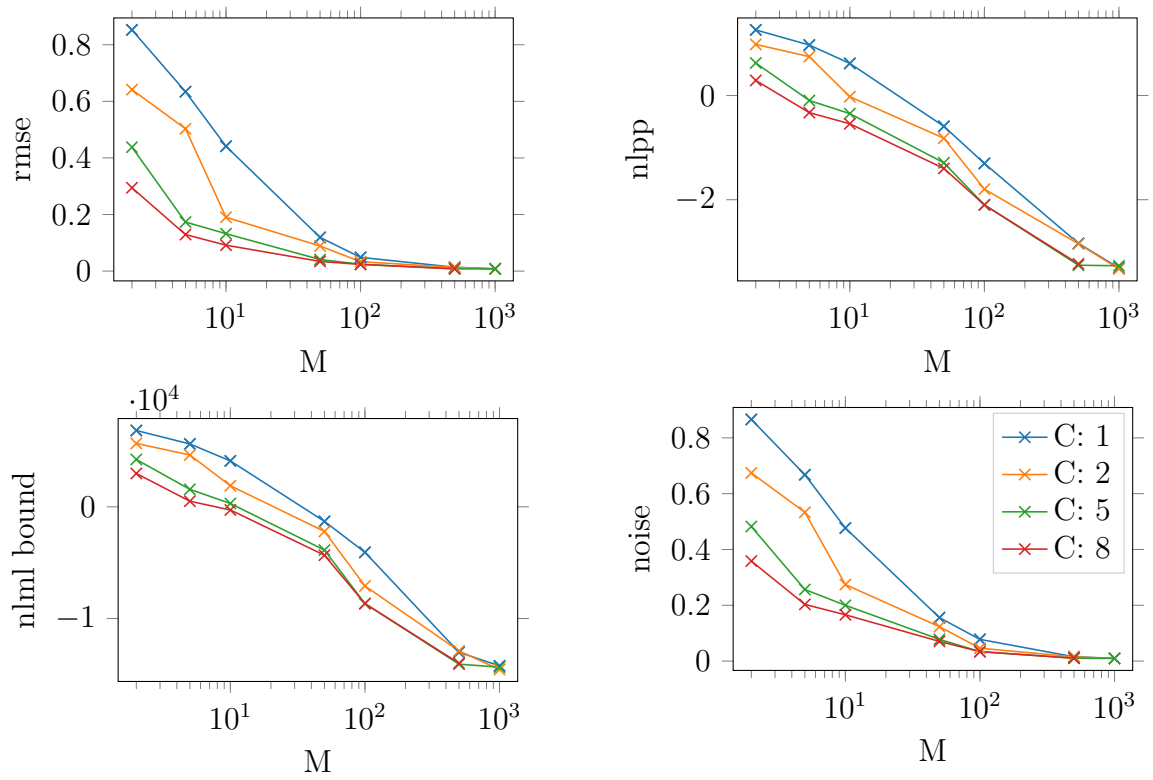


Fig. 4.7 Results for the Parkinsons dataset, showing performance in RMSE and NLPP (top left and right) and the negative bound on the log marginal likelihood (NLML) and estimated noise hyperparameter (bottom left and right).

In the NLML plot in figure 4.8 we see that we are starting to approach the regime where adding inducing points does not improve the bound. The results are broadly similar, with the added benefit of additional inter-domain basis functions diminishing.

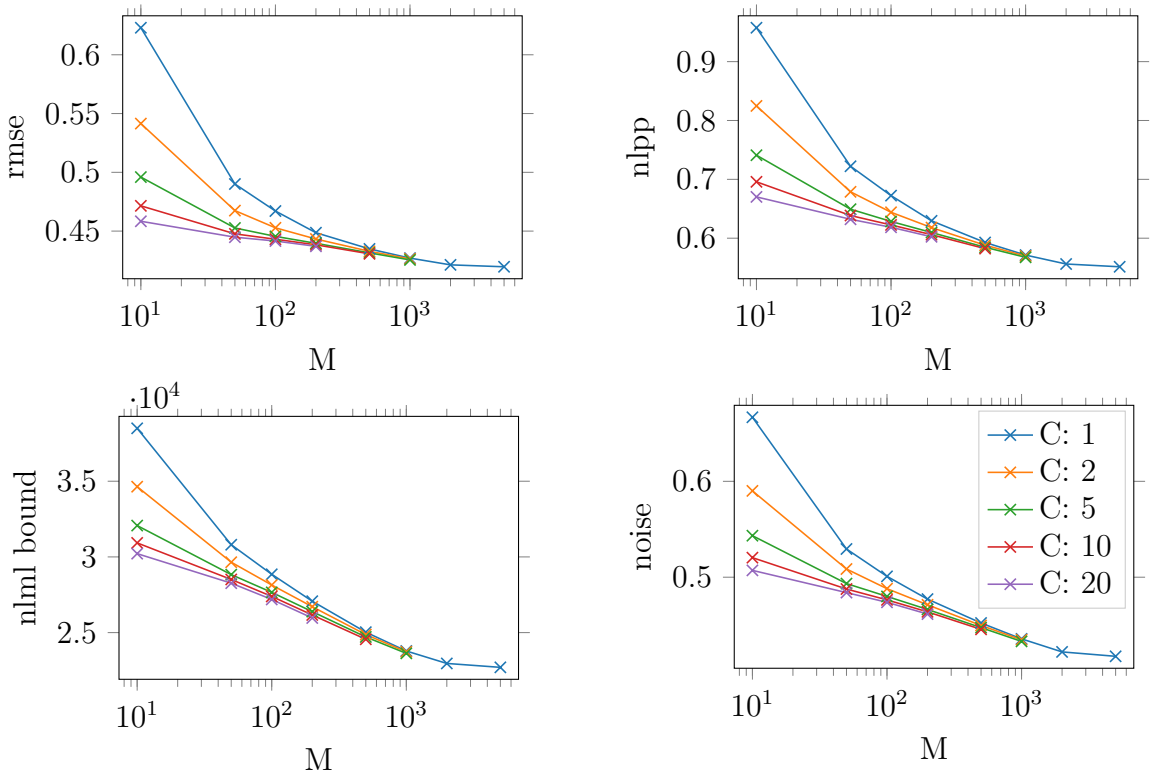


Fig. 4.8 Results for 36k split of kin40k, with 0.4 stddev noise added, showing performance in RMSE and NLPP (top left and right) and the negative bound on the log marginal likelihood (NLML) and estimated noise hyperparameter (bottom left and right).

Noisy Parkinsons

The result follow generally the same pattern as in for the non noisy version, only here it becomes clear that the inter-domain basis functions do help to achieve a tight bound with a lower sparsity.

4.5.2 Sparsity for known hyperparameters

In the previous section we saw that VFE had trouble capturing the true posterior for the kin40k dataset, even in the case with added noise that was designed to be easier. Once a certain number of "true" inducing variables are used, inter-domain basis

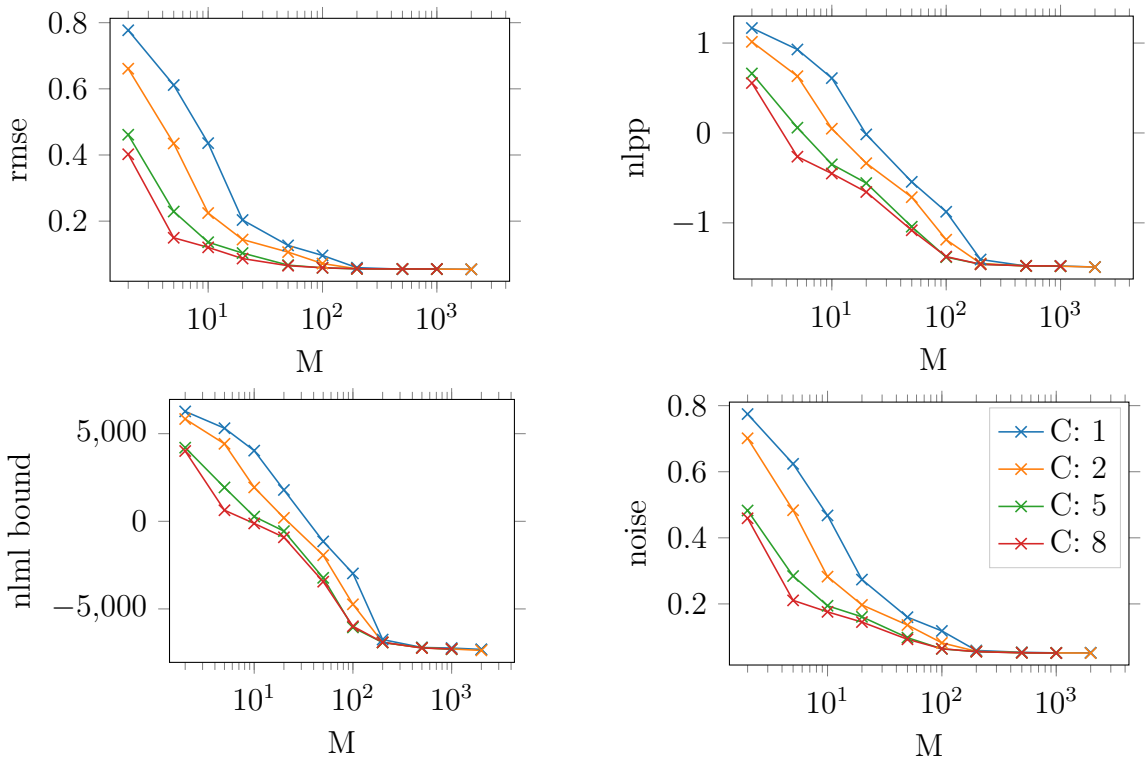


Fig. 4.9 Results for the Parkinsons dataset with 0.05 sttdev noise added, showing performance in RMSE and NLPP (top left and right) and the negative bound on the log marginal likelihood (NLML) and estimated noise hyperparameter (bottom left and right).

functions give diminishing returns to both the performance and the bound, despite the fact that the variational approximation has not captured the true posterior yet. This begs the question of why the extra flexibility of the inter-domain basis functions is not helpful and is not utilised.

One possibility is that the hyperparameters – particularly the noise – are biased due to the poor bound. Perhaps the extra inter-domain basis functions could be trained better if only there was a good estimate of the hyper-parameters? Here we investigate this by assessing how the inter-domain basis functions help when approximating a GP with given near-optimal hyperparameters, found by training a sparse model with $M = 10^4$ (i.e. impractically large).

In figure 4.10 we see that in terms of RMSE the inter-domain basis functions most definitely are utilised fully, and improve performance significantly. It seems that in this case inter-domain basis functions are equally valuable, with inter-domain basis functions being useful even at $M = 10$ with $C = 50$. However, this comes at the cost of performance in NLPP, where performance is worse than when the hyperparameters are optimised with the variational lower bound. Additionally, it is noteworthy how poor the estimate of the lower bound is.

4.5.3 Compression of the posterior

In previous sections we showed that we could indeed find sparser solutions in certain regimes using inter-domain basis functions, in the sense that fewer inducing variables were needed. Here we show that this sparsity can have practical benefits in terms of the size of the model that needs to be stored for predictions. This is a like-for-like comparison, which takes into account the extra parameters required by having multiple basis functions per inducing variable.

We assume a situation where we want to do the minimum amount of computation at test time. The form of the GP posterior is:

$$f(\cdot) \sim \mathcal{GP}\Big(k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}; k(\cdot,\cdot) - k_{\cdot\mathbf{u}}K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}} - \Sigma)K_{\mathbf{uu}}^{-1}k_{\mathbf{u}\cdot}\Big) \tag{4.20}$$

So for predictions we would need to store $K_{\mathbf{uu}}^{-1}\boldsymbol{\mu}$ (M elements), the matrix $K_{\mathbf{uu}}^{-1}$ $(K_{\mathbf{uu}} - \Sigma)$ $K_{\mathbf{uu}}^{-1}$ (M^2 elements), and the inducing inputs Z ($MC(D+1)$ elements) to allow the calculation of $k_{\mathbf{fu}}(\mathbf{x}^*, \mathbf{z})$. We see that inducing variables have a squared cost, while inter-domain basis functions only scale linearly, although scaled by the input dimension.

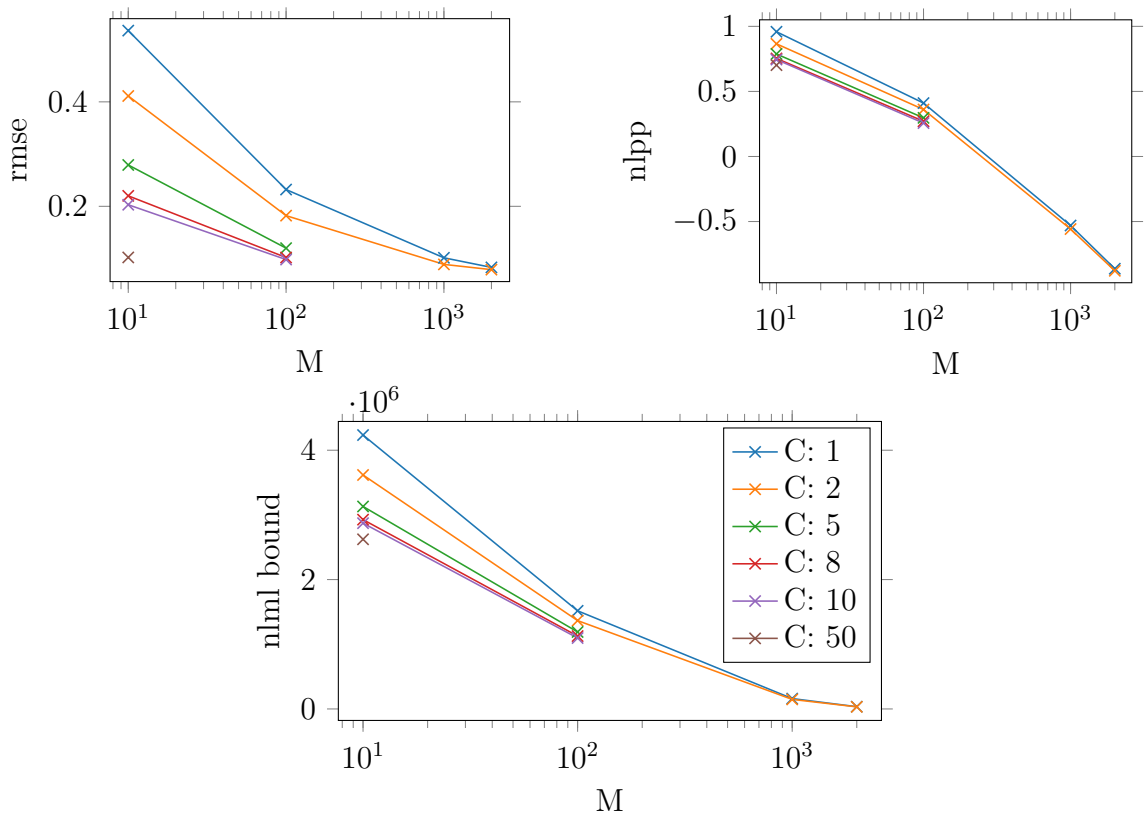


Fig. 4.10 Kin40k (36k split) trained with the hyperparameters fixed to good values found by using a sparse model with 10^4 inducing points.

Figures 4.11 and 4.12 show the relation between the size of the approximation and the obtained performance. Using inter-domain bases does compress models up to an order of magnitude, depending on the desired level of performance. As earlier, the benefit for kin40k decreases for large models when large M is required. A large improvement is more consistently seen in the Parkinsons dataset over a larger range of approximation sizes, with a consistent factor of 3 improvement in size for a chosen NLPP. In RMSE using inter-domain basis functions produces big gains for very sparse models. As all models become larger, the improvement in RMSE for an increase in model size diminishes. However, if one really requires to obtain a specific level of performance, inter-domain basis functions will still give compression factor of 2-3.

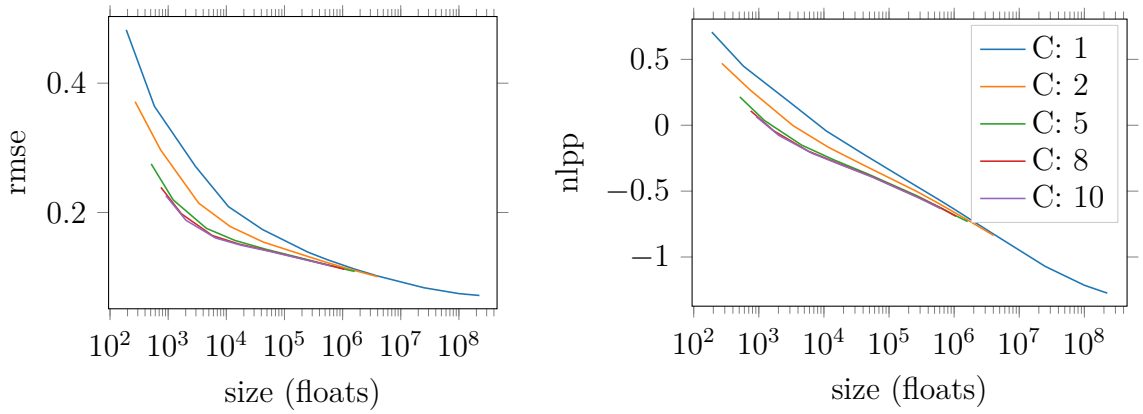


Fig. 4.11 Size of the kin40k model against performance.

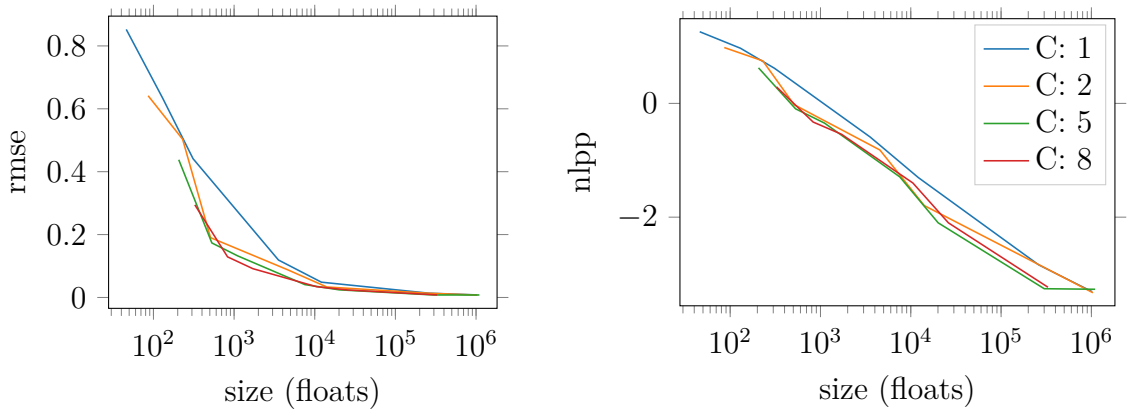


Fig. 4.12 Size of the Parkinsons model against performance.

4.5.4 Computational comparison

In section 4.4.1 we discussed that for calculating the bound and its gradients, the computational complexity of adding inter-domain bases is less than adding full inducing variables. Actual computational speed-ups depend on a multitude of factors, including details of the implementation. The current implementation is based on GPflow [de G. Matthews et al., 2017], a TensorFlow [Abadi et al., 2015] based library for Gaussian processes. Relying on TensorFlow for the computations has pushed the boundaries of practicality for GPs, firstly by removing the need for manual coding of derivatives, and secondly by more effectively utilising modern hardware like GPUs. However, this comes at the cost of some flexibility in implementation. The current implementation, for example, does not use the approach to minimise memory usage (section 4.4.1) – a boundary which was hit on the GPU. We believe there is still some scope for optimising the evaluation of the kernel matrices, which could change the results from this section.

Apart from implementation details, the nature of the optimisation problem is also changed with the introduction of more parameters. In order to separate the influence of the change in optimisation problem and iteration cost, we investigate the optimisation traces as a function of both iteration count and time.

In figure 4.13 we see the performance increase for a selected few models for the kin40k dataset. We see that when using inter-domain basis functions, more iterations are needed to reach optimal performance. While the inducing point models have all converged after roughly 2000 iterations, the models using inter-domain inducing variables are still slowly improving. For the kin40k dataset, the difference in iteration speed does not compensate for this difference.

The Parkinsons dataset (figure 4.14) shows similar features, with performance roughly the same.

4.6 Discussion

In the previous sections, we found that:

- Performance can be significantly improved for very sparse models by making the inter-domain variables more flexible. For some datasets the improvement can continue until the posterior is very close to optimal (Parkinsons dataset).

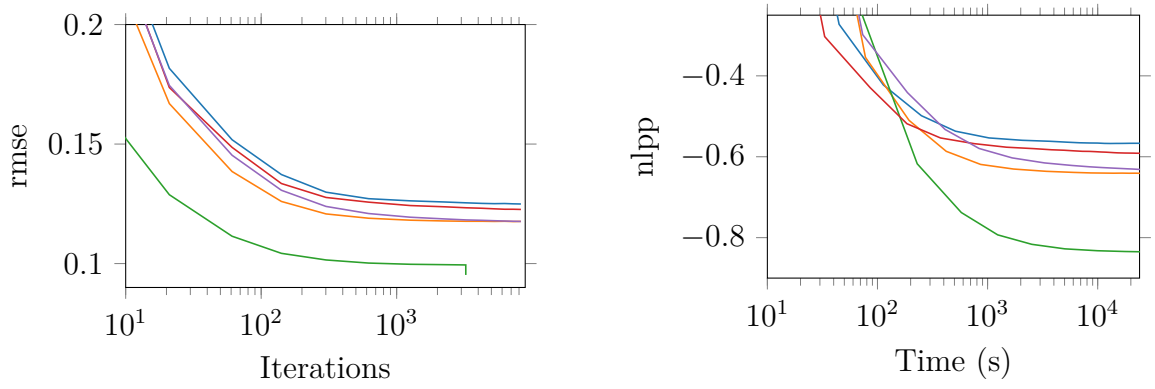


Fig. 4.13 RMSE and NLPP training curves comparing similarly performing models. Blue: $M = 750, C = 1$, orange: $M = 1000, C = 1$, green: $M = 2000, C = 1$, red: $M = 750, C = 2$, purple: $M = 750, C = 10$.

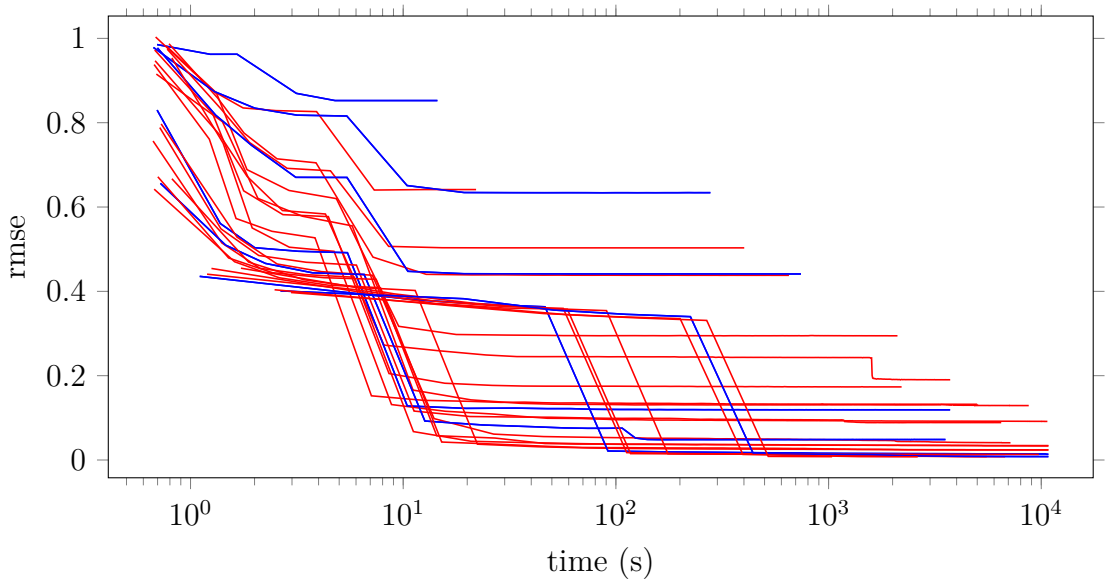


Fig. 4.14 Trade-off between run time and RMSE for the Parkinsons dataset. Blue lines show inducing point models, while red lines show models using inter-domain basis functions. While using inter-domain bases introduces a small improvement, the speed-up is probably not worth the additional complexity.

- In some cases the extra capacity allowed by the inter-domain basis functions is not utilised. This is caused by bias in the hyperparameter optimisation from the variational lower bound.
- Practically, inter-domain basis functions can be used to reduce the storage requirement for a given required level of performance.

We particularly want to note the result from section 4.5.2, and what this means for the viability of finding good GP approximations. This result shows that the variational bound may need a very large number of inducing *variables* (i.e. large M) to obtain an accurate posterior, while only 500 basis functions are really necessary to model the mean. The reason for this can be found by considering the form of the variational lower bound for regression in equation 3.1 together with the decomposition of the approximate posterior in figure 4.1. In section 3.2.1 we saw that the trace term will strongly bias the hyperparameters to large noise values if the inducing variables can not make the variance of $h(\cdot)$ small at data locations. Reducing this variance can only be achieved by increasing the rank of $Q_{\mathbf{ff}}$, which requires real inducing variables. Inter-domain basis functions allow the maximum amount of covariance to be squeezed out of a single inducing variable, but ultimately it will only impose a single linear constraint on the samples from the posterior.

Somehow it seems that using the variational bound makes the problem harder. Other models – parametric models – like the SSGP manage to predict much better with far fewer basis functions. Frustratingly, the SSGP is often justified as being an approximation to a GP with a given covariance function. Lázaro-Gredilla et al. [2010] and Gal and Turner [2015] both mention that stationary kernels can be approximated using sinusoidal features, and then move to optimising the features. While this definitely shows big gains in performance, this does not show that they approximate the posterior of the original model well. In fact, the fact that these models perform well is likely because they are *not* constrained to fit the original model. Section 4.5.2 shows that in a certain sense it is easy to learn the mean of the dataset, but hard to get the full posterior. Parametric models like the SSGP can freely adjust their prior to fit the data, while taking it way from the original model.

Does this mean it is foolish to try to approximate non-parametric posteriors? Krauth et al. [2016] seem to go down this path, showing that results can be improved by using an estimate of the Leave-One-Out objective, instead of the variational lower bound. This could significantly improve performance with inter-domain bases, since we know that we *can* obtain good performance with our chosen posterior. However, the

marginal likelihood for model selection, and the associated variances of the posterior are the unique things about Gaussian process models. Would this approach (to quote MacKay [1998]) be throwing out the baby with the bathwater?

To offer an alternative train of thought: Perhaps the problems we encountered with the kin40k dataset arise from the foolishness of trying to approximate the posterior of a *squared exponential* GP. If the kernel is mismatched to the dataset, and past points do not accurately predict future ones (resulting in a low marginal likelihood), we can't really expect to be able to find a good sparse approximation. Perhaps good models (i.e. with high marginal likelihood) allow for sparser approximations? Perhaps we should be searching over larger classes of models at the same time as finding approximate posteriors, in order to find models that we can both do inference in successfully, *and* that have high marginal likelihoods[2]?

4.7 Conclusion & Outlook

Overall, we have shown how to add cheaper basis functions to sparse posteriors, and experimentally shown that there are situations where they help, and where they make little difference. Practically, they can be used to store models more efficiently, without reducing performance. Additionally, we believe this work illustrates interesting properties of sparse GP posteriors, showing that the number of basis functions is *not* sufficient for good performance.

Hopefully inter-domain basis functions can be a stepping stone for improving sparse GP approximations. One problem that may hold them back is that they rely on optimisation for setting the weights of the inter-domain basis functions, something which the inducing variables get for free with the matrix inversion required for calculating the bound. Empirically, there is some evidence to show that this makes the optimisation problem harder. Even for regular inducing point methods, the optimisation of the inducing inputs is time consuming. In neural networks, much effort has gone into adjusting the models to be easy to optimise (ReLUs, Nair and Hinton [2010]), and developing optimisation algorithms [Grosse and Martens, 2016]. Perhaps the GP community would benefit from the same.

[2] A similar thought is mentioned by Huszár [2016].

Chapter 5

Convolutions and Invariances

In the previous sections, we mainly investigated how to come up with accurate approximations to the posterior of a given model, when the dataset is too large to allow exact computations. However, accurate inference is only useful so far as the model in question is able to generalise well. Until now, we haven't paid much attention to the properties of the GP prior, and how these affect its ability to generalise well in a specific problem. Most common kernels rely on rather rudimentary and local metrics for generalisation, like the Euclidean distance. Bengio [2009] has argued that this has held back kernel methods in comparison to deep methods which learn more sophisticated distance metrics that allow for more non-local generalisation. While deep architectures have seen enormous success in recent years, it is an interesting research question to investigate what kind of non-local generalisation structures *can* be encoded in shallow structures like kernels, so that the elegant properties of GPs are preserved. Additionally, a more complete understanding of the abilities and limits of kernels, may help improve the understanding of what sets deep models apart, or make kernel methods competitive once again.

In this chapter, we introduce a Gaussian process with convolutional structure, together with a matched approximate inference scheme that makes the method applicable in practice. Our method distinguishes itself from previous attempts to introduce convolutions to GPs by convolving a Gaussian process over the inputs, rather than by mapping the inputs through a deterministic transformation like a neural network [Calandra et al., 2016; Wilson et al., 2016]. We view convolutional structure in the light of more general invariances [Kondor, 2008; Ginsbourger et al., 2013; Duvenaud, 2014], to which our method is also applicable, with the aim of developing a general understanding of why invariances are useful. We demonstrate the method on MNIST and CIFAR-10.

Overall, this method is a significant step towards bringing the advantages of Gaussian processes to convolutional networks. Thanks to the variational lower bound, we have an automatic way of choosing how many filters to use, how to choose hyperparameters and even what model structure to choose. In the coming sections we will first review why invariances in the kernel may help generalisation, followed by how our approximation can be used for general invariances. We then introduce the convolutional kernel and some of its variants and show its use experimentally.

This section is joint work with James Hensman, and was published in van der Wilk, Rasmussen, and Hensman [2017].

5.1 Improving generalisation through invariances

We now shift our focus from approximating GP models with a given kernel to designing and identifying models which will perform well on a particular dataset. Any statistical model has to make assumptions about how the observed data relates to future predictions in order to be able to make any predictions, and the success of a method depends on the suitability of the assumptions for the dataset in question. In Bayesian modelling (section 1.1) we express all assumptions of a model in its prior and likelihood. We can propose multiple models, and use the marginal likelihood to assess the suitability of each of the assumptions made. As discussed, the predictions when averaging over all models usually are dominated by a single model with the largest marginal likelihood.

A central expectation in the Bayesian methodology is that models with a high marginal likelihood will generalise well. There are many strands of work that quantify generalisation error in GP models, and make the link to marginal likelihoods [Seeger, 2003; Germain et al., 2016][1]. So how do we obtain Gaussian processes that have high marginal likelihoods?

Since the properties of a GP prior are fully determined by its kernel, finding a good GP prior is equivalent to finding a good kernel. The marginal likelihood can be used to choose between continuously parameterised kernels, for example by using gradient-based maximisation with respect to lengthscales and variance parameters of simple kernels like the squared exponential. More flexible, highly parameterised kernels can also be used, like the spectral mixture kernel [Wilson and Adams, 2013], or kernels parameterised by neural networks [Wilson et al., 2016]. Comparison between discrete models is also possible, as was done by Duvenaud et al. [2013]. Here, the marginal

[1] Estimating or bounding generalisation error directly can also be used for assessing models – see Rasmussen and Williams [2005, ch.5] for an introduction and pointers.

likelihood is used to perform a greedy search over a discrete space of kernel structures to create the "Automatic Statistician".

All methods above essentially search over a large space of kernels using the maximum marginal likelihood. The larger these search spaces are, the less constrained the kernel hyper-parameters are, which leads to a loss in robustness to over-fitting. Extra regularisation is often needed. In the next few sections, we investigate what characteristics in a GP prior gives a model a high marginal likelihood, and how to design kernels with these characteristics, without necessarily choosing from a large parametric family. We will specifically investigate invariances.

5.1.1 Model complexity and marginal likelihoods

The approaches mentioned in the previous section all search for kernels by maximising the marginal likelihood (justified in section 1.1.4). But what properties do kernels with high marginal likelihoods have? In this section, we analyse the marginal likelihood further to gain an insight into what functions should have high probability under the prior to allow for high marginal likelihoods. We follow MacKay [2002, ch.28] and Rasmussen and Ghahramani [2001] and argue that the marginal likelihood prefers models which are as constrained as possible, while still fitting the data. This justifies our later investigation into incorporating invariances and convolutional structures into kernels in order to improve generalisation.

The marginal likelihood of a model is found by evaluating the probability it assigns to generating the observed data. It is usually expressed as the marginal of the likelihood and prior (hence the name):

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y}|f)p(f|\theta)\mathrm{d}f\,. \tag{5.1}$$

This formulation invites us to gain intuition by considering sampling from the prior. We can think of a Monte Carlo approximation to the marginal likelihood by sampling functions from the prior and then averaging the likelihood of each sample. If many functions from the prior give a high $p(\mathbf{y}|f)$, we obtain a high marginal likelihood.

We visualise this in figure 5.1, where we show samples from three different priors $p(f)$ with increasing lengthscales, and the samples left over after we filter out the ones with low likelihood $p(\mathbf{y}|f)$. The number of high likelihood samples is a proxy for the marginal likelihood. Functions from the short lengthscale prior vary too much. If a function passes through one particular data point, there is only a small chance it will also pass through a neighbouring one. However, this does allow the model to assign

similar marginal likelihoods to many data sets. Taking this model to the extreme by letting the lengthscale go to zero, we obtain a white noise process which assigns equal probability to all (normalised) datasets. The bottom example shows a prior with too long a lengthscale. Functions are constrained from varying between nearby points, resulting in a low probability of the function changing enough to be pass through neighbouring observations which differ. The limit of taking the lengthscale to infinity gives a model that only produces flat, horizontal functions. Such a model will give poor marginal likelihoods for all datasets, except ones where all observations lie on a horizontal line. In between these two models, is an optimal model, which contains functions with just the right amount of flexibility.

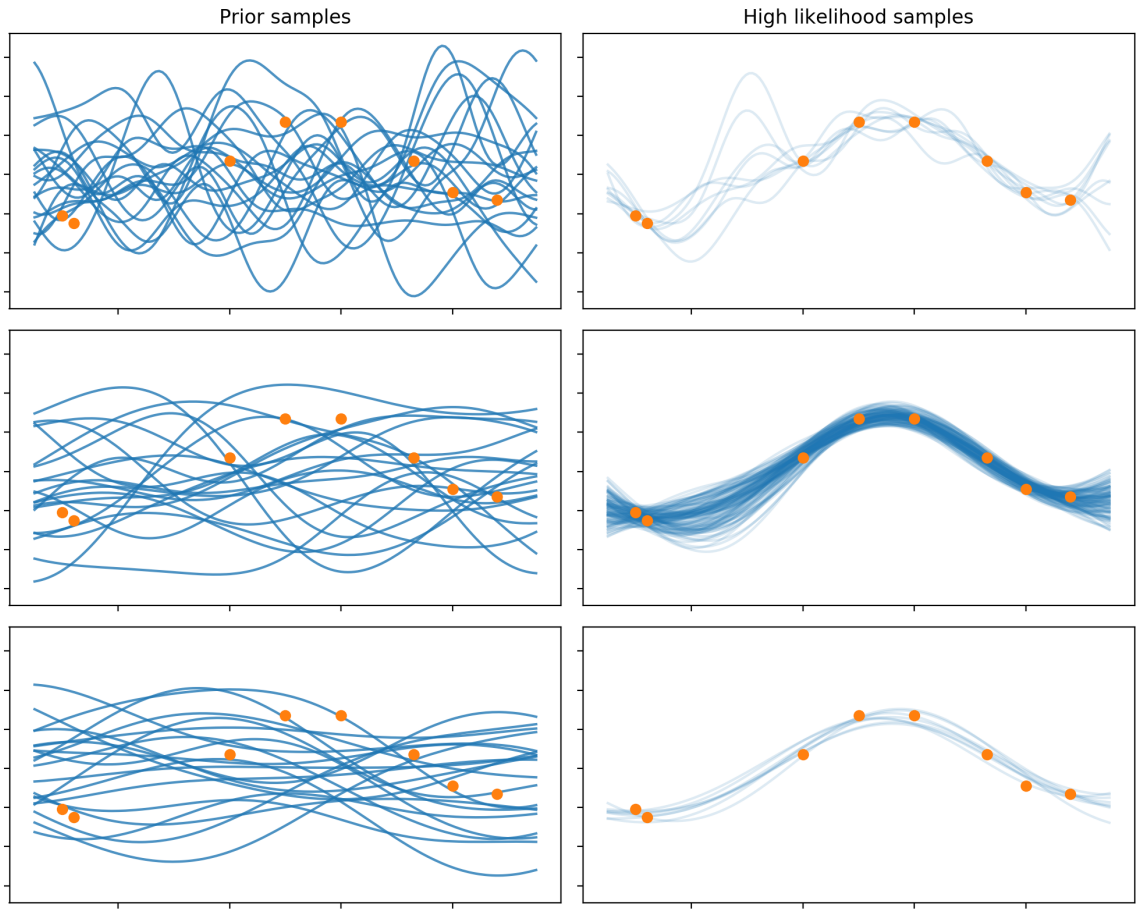


Fig. 5.1 Samples from priors with increasing lengthscales (left), and the samples from the prior that have high likelihood. From the 10^6 samples from the prior, only 9, 120, and 8 have high likelihood for the respective models.

We can also use the product rule to factorise the marginal likelihood, rather than writing it as a marginal over f. This is interesting, as it shows that the marginal

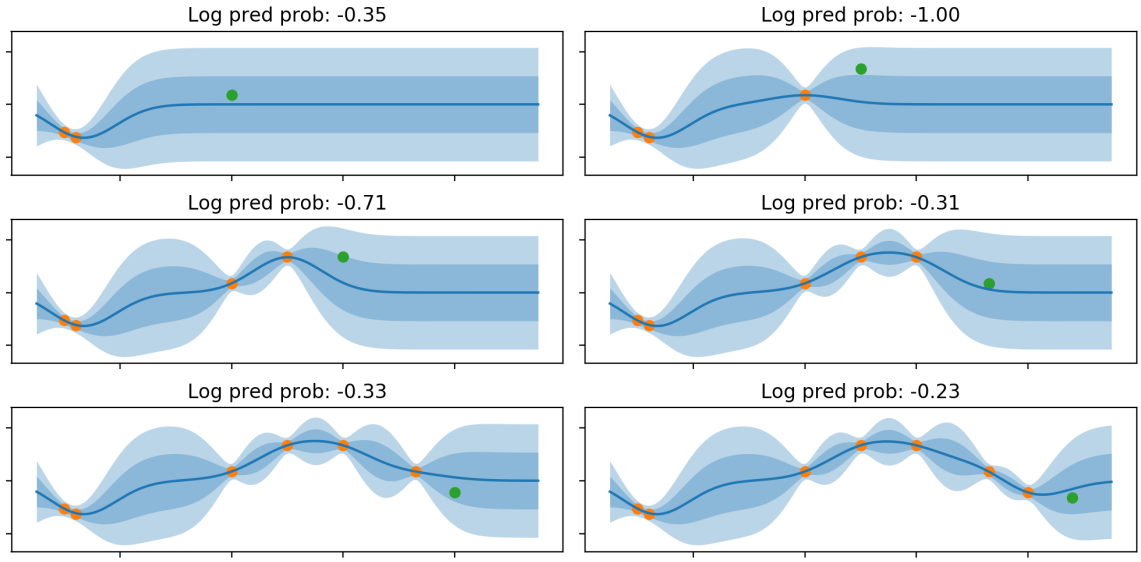


Fig. 5.2 Short lengthscale prior's incremental predictive probabilities.

likelihood can be written in terms of the product of posterior predictive distributions of the nth point, given all previous observations.

$$\begin{aligned} p(\mathbf{y}|\theta) &= p(y_1|\theta)p(y_2|y_1,\theta)p(y_3|y_{1:2},\theta)p(y_4|y_{1:3},\theta)\ldots \\ &= p(y_1|\theta)\prod_{n=2} p(y_n|y_{1:n-1},\theta)\,. \end{aligned} \tag{5.2}$$

We visualise this in figures 5.2 to 5.4. The short lengthscale prior (figure 5.2) predicts with large variances, which fits with the earlier intuition of the prior not being constrained enough. Even conditioned on several observations, the prior still allows enough variation in the functions for the prediction at the new point to be uncertain. The prior with the long lengthscale (figure 5.4) is again over-constrained. The prior does not allow for much variation within each sampled function, so the model makes very strong predictions even after only a few observations. Since the observed data lie far outside the high density region of the prediction, the marginal likelihood incurs a strong loss. Again, the medium lengthscale prior is constrained such that it predicts with the narrowest error bars, while keeping the data within them.

5.1.2 Invariances help generalisation

In the previous section, we only changed the lengthscale of a kernel to constrain the flexibility of functions in the prior. This is a rather rudimentary constraint. Intuitively,

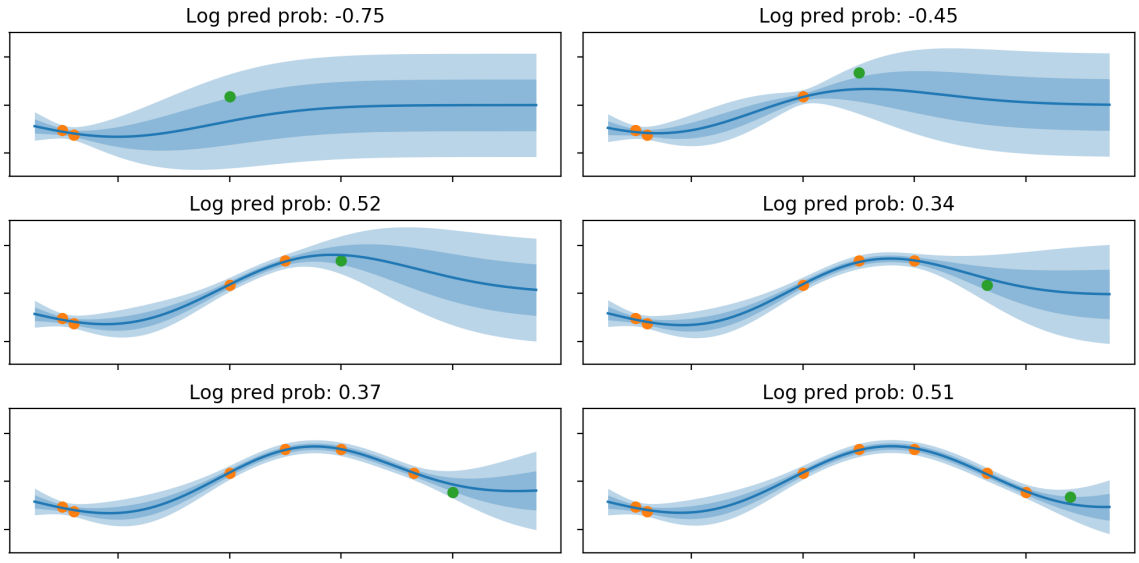


Fig. 5.3 Medium lengthscale (good fit) prior's incremental predictive probabilities.

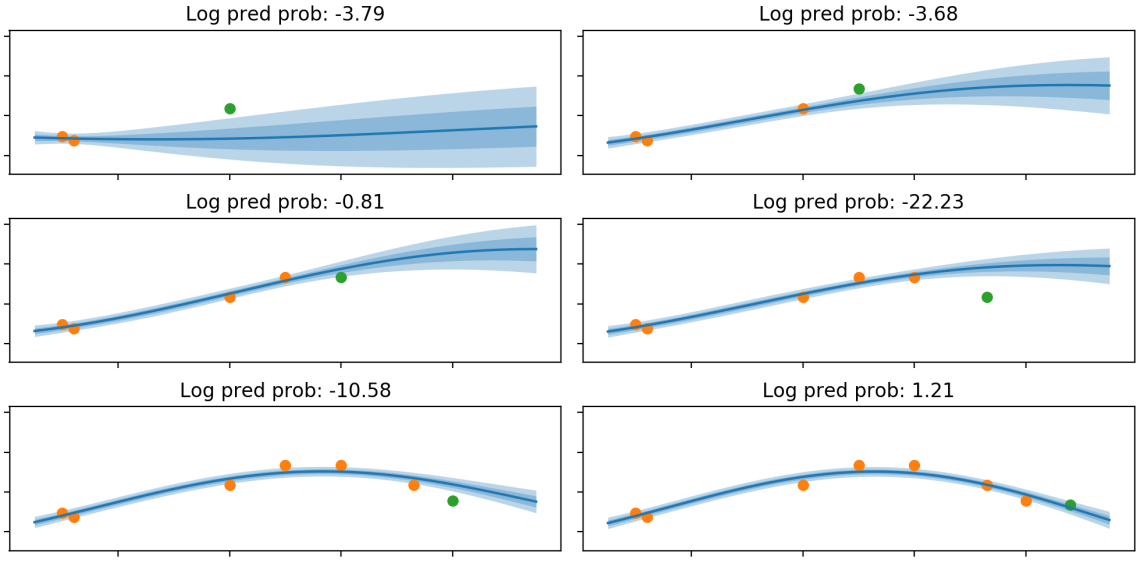


Fig. 5.4 Long lengthscale prior's incremental predictive probabilities.

we want a machine learning method to predict the same (or a similar) value for inputs which are similar in our abstract perception of the inputs (e.g. all ways of writing a particular digit share some essential characteristics). Kernels based on the Euclidean distance will only constrain variation with respect to this metric, which is well known to not match humans' perception of variation in many perceptual tasks. Instead, we investigate constraining all functions in a prior to be invariant to some transformation. Invariances are a strong constraint, and greatly help generalisation – so long as they are actually present in the data. The periodic kernel [MacKay, 1998], for example, only contains functions in the prior that are periodic with a specified period. With this extra constraint, observations inform what the function does even in regions that are distant in terms of the euclidean distance (figure 5.5). Thinking back to equation 5.2, we can see how this gives a possibility for higher marginal likelihoods that did not exist with the squared exponential kernel. The tight predictions in distant regions have the opportunity to contribute to a high marginal likelihood.

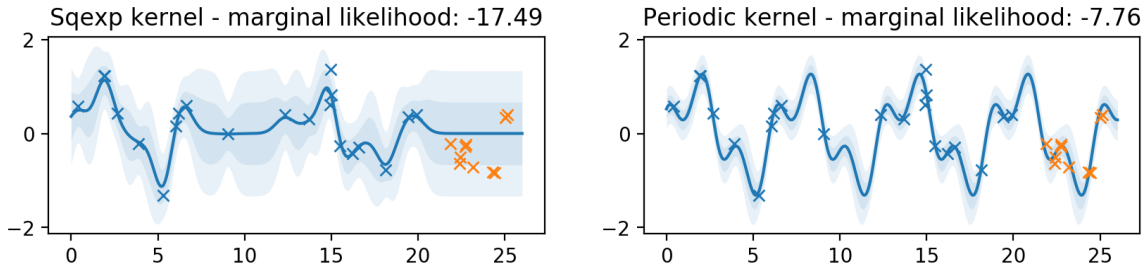


Fig. 5.5 Regression on a periodic function. The squared exponential kernel leads to a prior where with functions that are locally smooth. Function values far away from observations are largely unconstrained, leading to uncertainty when extrapolating. A prior constrained to contain only periodic functions (i.e. ones which are invariant to translations by the period) extrapolates much better for this example that actually does exhibit periodicity.

We are particularly interested in high dimensional inputs like images, where constraining the model is crucial due to the many degrees of freedom that functions can have in these spaces (the curse of dimensionality). Consider a squared exponential kernel which uses the Euclidean distance to determine the decay in covariance between points, and say the function is known at some points in a small region. Since the squared exponential kernel allows the function to vary independently in any direction away from the data, the GP will quickly revert back to the prior as we move in any direction away from a training point. As a consequence, the posterior will likely predict with large uncertainties on test data. In order to learn in high dimensional spaces,

a kernel must effectively be able to rule out variation of the function in the correct directions.

Exploiting additive structure in kernels [Durrande et al., 2012; Duvenaud et al., 2011] is one way to reduce the flexibility of the prior functions in high dimensions. The simplest additive Gaussian process has a kernel which is additive over dimensions (e.g. pixels of an image)

$$k(\mathbf{x}, \mathbf{x}') = \sum_i k_0(x_i, x_i')\,, \tag{5.3}$$

where $k_0(\cdot, \cdot)$ is a kernel over 1 dimensional inputs, and x_i is the ith element of $\mathbf{x}$. The additive structure enforces that all resulting functions are sums of functions of one dimension. This causes the *relative* values of the function to be invariant to movement along one dimension. As a consequence, if an offset is learned at one location, the same offset will be applied along all other dimensions. This introduces long range correlations that simply do not exist in common kernels like the squared exponential, where the influence drops off equally quickly in all directions.

Other invariances share this same quality of long range interactions. For example, if we learn a function that is constrained to be invariant to rotations or translations of the an image input, a single observation will constrain the function for all images that are rotated / translated versions, which will be far away as measured by the Euclidean distance.

5.2 Encoding invariances in kernels

We discussed that invariances can be helpful to constrain models in high dimensional spaces in a way which (hopefully) helps them to generalise. So, how can these invariances be encoded into GP priors? Kondor [2008] and Duvenaud [2014, §2.7] provide great discussions from the machine learning point of view, and we will review one particular method here.

Invariance of a function $f(\cdot)$ can be formalised as a requirement that for a collection of T transformations of the input space of the GP, $g_i : \mathcal{X} \to \mathcal{X}$, we have:

$$f(\mathbf{x}) = f(g_i(\mathbf{x})) \qquad \forall \mathbf{x} \in \mathcal{X} \qquad \forall i \in \{1 \dots T\}\,. \tag{5.4}$$

The composition of all compositions of these operations forms a *group* G containing every possible transformation that we are required to be invariant to. Ginsbourger

et al. [2013] showed that a GP prior produces samples with the required invariances if and only if the kernel is invariant to the same transformations:

$$k(\mathbf{x}, \mathbf{x}') = k(g(\mathbf{x}), g'(\mathbf{x}')) \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} \qquad g, g' \in G\,. \tag{5.5}$$

One way to construct such a kernel, as suggested by Kondor [2008] and Ginsbourger et al. [2013], is to sum a base kernel over the *orbit* of $\mathbf{x}$ with respect to the group of G. The orbit of $\mathbf{x}$ is the set of points that one obtains by applying all transformations in G to $\mathbf{x}$: $\{g(\mathbf{x}) \mid g \in G\}$. We obtain the kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{g \in G} \sum_{g' \in G} k_{base}(g(\mathbf{x}), g'(\mathbf{x}'))\,. \tag{5.6}$$

We can (informally) justify this approach by noting that this is exactly the covariance obtained if we construct a function $f(\mathbf{x})$ by summing some base function $f_{base}(\mathbf{x}) \sim \mathcal{GP}(0, k_{base}(\cdot, \cdot))$ over the orbit of $\mathbf{x}$:

$$f(\mathbf{x}) = \sum_{g \in G} f_{base}(g(\mathbf{x}))\,. \tag{5.7}$$

Any two points that can be transformed to each other will have the same orbit, causing their respective function values to be equal, as they are constructed from a sum of the same values. The kernel for $f(\cdot)$ can be obtained by taking expectations over $f_{base}(\cdot)$:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{f_{base}}\left[\sum_{g \in G} f_{base}(g(\mathbf{x})) \sum_{g' \in G} f_{base}(g'(\mathbf{x}'))\right] \\ &= \sum_{g \in G} \sum_{g' \in G} \mathbb{E}_{f_{base}}[f_{base}(g(\mathbf{x}))f_{base}(g'(\mathbf{x}'))] \\ &= \sum_{g \in G} \sum_{g' \in G} k_{base}(g(\mathbf{x}), g'(\mathbf{x}')) \end{aligned} \tag{5.8}$$

This result is really convenient, as invariances are encoded simply as kernels, which can be evaluated and plugged in to normal Gaussian process inference code.

5.2.1 Computational issues

While the previous section provides a constructive procedure for coming up with an invariant kernel, computational resources can once again pose a limit – and for different reasons than are usual for Gaussian processes. Denoting the size of the orbit as P, a single evaluation of the invariant kernel requires P^2 kernel evaluations. If the size

of the orbit is large, this may make the kernel evaluation intractable, on top of the intractability of the large matrix inverse. Taking the kernel evaluations into account, the full GP has the cost $\mathcal{O}(N^3 + N^2P^2)$, while the VFE bound (equation 2.36) costs $\mathcal{O}\left(\tilde{N}M^2 + \tilde{N}MP^2 + M^2P^2\right)$.[2]

If we consider a kernel that is invariant to pixel-wise translations of the input images, a problem of the size of MNIST would have $P = 27^2 = 729$ (similar for the convolutional kernels introduced later). Even in sparse methods, the $6 \cdot 10^5$ kernel evaluations required *per* entry in $K_{\mathbf{fu}}$, $K_{\mathbf{uu}}$ and $\text{diag}[K_{\mathbf{ff}}]$ are completely prohibitive[3]. When using 750 inducing variables (probably still not enough), storing $K_{\mathbf{uu}}$ alone would take more than 1.1TB of memory. And this is supposed to be the tractable matrix! A better approximation is needed.

Ultimately, this intractability stems from the approximate posterior we chose in section 2.4.1. Although it can be manipulated without any large, intractable matrix inverses, it still requires evaluations of the kernel, which in this case is intractable due to its double sums. Any method which uses this class of approximate posteriors will suffer from the same problem. This indicates that we should first search for a tractable approximate posterior that can be used with invariant kernels. If this is possible, the solution should be applicable to both VFE and FITC/EP.

5.2.2 Inter-domain variables for invariant kernels

In chapter 4 we discussed *inter-domain* inducing variables, where we place the inducing inputs in a different space than our observations and predictions. The original work by Lázaro-Gredilla and Figueiras-Vidal [2009] focused on using integral transforms of $f \sim \mathcal{GP}$ to define the space where the new inducing inputs would lie. In Matthews's [2016] later measure-theoretic treatment of sparse Gaussian processes, it was shown that any deterministic relationship between the inducing domain process $u(\mathbf{z})$ and the GP of interest $f(\mathbf{x})$ would result in a method that minimised only the KL divergence between approximate and true posterior processes. Here, we construct a sparse inter-domain approximation for invariant kernels of the form of equation 5.6, which avoids the prohibitive $\mathcal{O}(P^2)$ cost.

The key observation that leads to our method is that in equation 5.7, we construct the invariant function by summing a base function over many transformed inputs. Given the deterministic relationship between $f(\cdot)$ and $f_{base}(\cdot)$, we choose to place the

[2] With minibatches of size $\tilde{N}$. Full-batch methods have $\tilde{N} = N$.

[3] The same issues regarding back-propagating gradients as mentioned in section 4.4.1 apply here as well.

inducing variables in the input space of f_{base}. We can find the (cross-)covariances as in equation 5.8.

$$k(\mathbf{x}, \mathbf{z}) = \mathrm{Cov}_{f_{base}}[f(\mathbf{x}) f_{base}(\mathbf{z})] = \mathrm{Cov}_{f_{base}}\left[\sum_{g \in G} f_{base}(g(\mathbf{x})) f_{base}(\mathbf{z})\right]$$
$$= \sum_{g \in G} k_{base}(\mathbf{x}, \mathbf{z}) \tag{5.9}$$
$$k(\mathbf{z}, \mathbf{z}') = k_{base}(\mathbf{z}, \mathbf{z}) \tag{5.10}$$

These inter-domain variables remove the double sum in the covariances needed for the computation of $K_{\mathbf{fu}}$ and $K_{\mathbf{uu}}$ – the largest kernel matrices. Only the computation of $\mathrm{diag}[K_{\mathbf{ff}}]$ in the variational bound still requires P^2 kernel evaluations, making the cost for the sparse method $\mathcal{O}\left(\tilde{N}M^2 + \tilde{N}MP + M^2 + \tilde{N}P^2\right)$. Luckily, for minibatch methods $\tilde{N}$ can be small compared to the dataset size, or even the number of inducing variables.

The use of inter-domain variables for parameterising the approximate posterior successfully addresses the computational constraints discussed in the previous section. Additionally, they are easy to implement within all sparse GP approximation frameworks (including both VFE and FITC/EP), as only a drop-in replacements for the calculation of $K_{\mathbf{fu}}$ and $K_{\mathbf{uu}}$ are needed. One question which is currently unanswered is whether these new inter-domain inducing variables are as effective as inducing points. This trick will not be particularly useful if the number of inducing variables required to obtain an approximation of a particular quality would drastically increase. We will address this question for a special case in section 5.3.2.

5.3 Convolutional Gaussian processes

Incorporating invariances into models to improve generalisation is ubiquitous in modern machine learning in the form Convolutional Neural Networks [LeCun et al., 1998]. Here, we describe an analogous Convolutional Gaussian process, which is constructed in a similar way to the invariant kernels of the previous section. In the next few sections, we will introduce several variants of the Convolutional Gaussian process, and illustrate its properties using synthetic and real datasets. Our main contribution is showing that convolutional structure can be embedded in kernels, and that they can be used within the framework of non-parametric Gaussian process approximations. For inference, we use a slightly modified version of the trick described in the previous section.

We are interested in the problem of constructing Gaussian process priors that generalise well for images in $\mathbb{R}^D$. As mentioned earlier, one issue with common kernels like the squared exponential, is that they are, in a certain sense, too flexible. Additive models allow the value of a particular subset of inputs to influence the prediction far away in other inputs. Fully connected neural networks face a similar issue. Convolutional Neural Networks [LeCun et al., 1998] constrain the flexibility of the mapping by convolving many small linear filters with the input image, followed by a non-linearity. The main justification for this is that in images, small patches can be highly informative of the image label, and that the location of these patches matters less than their presence. It therefore makes sense to learn the same local features for the entire image.

Here, we construct a Gaussian process that has a similar structure by decomposing the input into a set of patches, and then applying the *same* function to each of these patches. The individual *patch responses* are pooled by an addition. This structure would allow each patch to contribute a particular value to the GP output and reduce its variance by some amount, independent of all other patches. The resulting kernel will be invariant to any image which contains the same patches, regardless of their location. We will also discuss modifications that remove the strict invariance constraint, in favour of extra flexibility.

Pandey and Dukkipati [2014] and Mairal et al. [2014] discuss similar kernels in passing. Computational constraints have prevented these ideas from being applied to GPs directly.

5.3.1 Constructing convolutional kernels

We construct the convolutional GP by starting in a similar way to equation 5.6 and equation 5.7. However, since we are interested in invariances over *patches* rather than the entire image, we choose f_{base} to act on smaller patches instead. This has the added advantage that we are effectively learning a Gaussian process of a smaller dimensionality, as well as any invariance benefits. This presents a trade-off for the model. Large patches allow structure between distant pixels to be distinguished, while small patches mitigate the curse of dimensionality.

We call our $f_{base}(\cdot)$ the *patch response function*, and denote it as $g : \mathbb{R}^E \to \mathbb{R}$, mapping from patches of size E. For images of size $D = W \times H$, and patches of size $E = w \times h$, we get a total of $P = (W - w + 1) \times (H - h + 1)$ patches. We can start by simply making the overall function from the image $f : \mathbb{R}^D \to \mathbb{R}$ the sum of all patch

responses. If $g(\cdot)$ is given a GP prior, a GP prior will also be induced on $f(\cdot)$:

$$g \sim \mathcal{GP}(0, k_g(\mathbf{z}, \mathbf{z}'))\,, \tag{5.11}$$

$$f(\mathbf{x}) = \sum_p g(\mathbf{x}^{[p]})\,, \tag{5.12}$$

$$\implies f \sim \mathcal{GP}\left(0, \sum_{p=1}^{P}\sum_{p'=1}^{P} k_g\left(\mathbf{x}^{[p]}, \mathbf{x}'^{[p']}\right)\right), \tag{5.13}$$

where $\mathbf{x}^{[p]}$ indicates the p^{th} patch of the vector $\mathbf{x}$. In the coming sections we will consider several variants of this model, particularly ones that remove the restriction of each patch contributing equally to f, and ones designed to operate on colour images.

5.3.2 Inducing patch approximations

The structure of this kernel is exactly that of invariant kernels (equation 5.6), and we can therefore apply the inter-domain approximation from section 5.2.2 by placing the inducing points in $g(\cdot)$:

$$u_m = g(\mathbf{z}_m)\,, \qquad\qquad \mathbf{u} = g(Z)\,. \tag{5.14}$$

One interesting detail is that the inducing inputs are now of a different (smaller!) dimensionality to the image inputs, as they live in patch space. The reduced dimensionality of the *inducing patches* may also make optimisation more manageable. The (cross-)covariances we need for parameterising the approximate posterior are in essence the same as equations 5.9 and 5.10, only summing over the patches in the image, rather than an orbit:

$$k_{fu}(\mathbf{x}, \mathbf{z}) = \mathbb{E}_g\left[\sum_p g(\mathbf{x}^{[p]})g(\mathbf{z})\right] = \sum_p k_g\left(\mathbf{x}^{[p]}, \mathbf{z}\right), \tag{5.15}$$

$$k_{uu}(\mathbf{z}, \mathbf{z}') = k_g(\mathbf{z}, \mathbf{z}')\,. \tag{5.16}$$

As pointed out earlier, this parameterisation clearly removes some of the computational issues with evaluating $K_{\mathbf{fu}}$ and $K_{\mathbf{uu}}$. However, this trick would not be of much use if the number of inducing variables that are required for a given approximation accuracy would increase drastically. In the next section, we investigate to whether this is an issue, and provide results and intuition into when the inducing patch approximation is accurate.

5.4 Required number of inducing patches

For inducing point approximations, it is well known when the exact posterior is recovered, and there is good intuition for when the approximate posterior will be accurate. Here, we prove similar results for inducing patch approximations, starting with what inducing patches are needed for an optimal approximation, followed by a characterisation of when this is more or less than what is needed when using inducing points. While this does not strictly tell us which approximation will be best when the number of inducing variables is constrained, a similar intuition holds for inducing patches, as for inducing points. Here, we will prove statements for the finite dimensional KL divergences, which are equal to the full KL divergence between processes [Matthews, 2016]. In the following, we will only refer to inducing variables corresponding to patch inputs as $\mathbf{u}$, while denoting inducing points as observations on $f(\cdot)$ directly.

5.4.1 Inducing points

It is well known that, for Gaussian likelihoods, the variational approximation and most other inducing point methods recover the true posterior when the inducing inputs are chosen to be the entire training input set $Z = X$ [Snelson and Ghahramani, 2006; Quiñonero-Candela and Rasmussen, 2005; Titsias, 2009b]. For non-Gaussian likelihoods with a full rank Gaussian $q(f(Z))$, the variational bound can always be improved when new inducing points are added [Matthews, 2016]. We summarise and prove these results in the following lemma.

Lemma 1. *For general, point-wise likelihoods $p(y_n|f(\mathbf{x}_n))$, the optimal placement for inducing points Z is at the training points X.*

Proof. We consider the KL divergence to an approximate posterior which uses the training inputs X and an arbitrary set of extra inducing points Z^* as inducing points, i.e. $Z = X \cup Z^*$. We denote the KL divergence between processes for the inducing point approximation as D_f, which evaluates to:

$$D_f = \mathrm{KL}[\, q(f(Z^*), f(X)) \,\|\, p(f(Z^*), f(X)|\mathbf{y}) \,]\,. \tag{5.17}$$

By applying the chain rule, we can decompose this expression into the KL divergence using only X as inducing points, and a positive term:

$$D_f = \underbrace{\mathrm{KL}[\, q(f(X)) \,\|\, p(f(X)|\mathbf{y}) \,]}_{\text{KL without additional } Z^*} + \underbrace{\mathbb{E}_{q(f(X))}[\mathrm{KL}[\, q(f(Z^*)|f(X)) \,\|\, p(f(Z^*)|f(X)) \,]]}_{\geq 0} . \tag{5.18}$$

Therefore, the KL divergence can only be increased by adding inducing points on top of X, with no extra gap being added if the conditional $q(f(Z^*)|f(X))$ is chosen to be equal to the prior conditional. □

Due to the highly structured nature of the convolutional kernel defined in equation 5.13, fewer inducing points are needed in some situations. Additionally, in some situations, the covariance for our Gaussian $q(f(Z))$ must be constrained to avoid a $-\infty$ variational lower bound. Before we summarise the result in the next lemma, we set up some useful notation. We denote with $\mathbf{u} \in \mathbb{R}^U$ the vector of patch response function values at the union of all patches present in all input images as

$$\mathbf{u} = g\left(\bigcup_{n=1}^{N} \left\{ \mathbf{x}_n^{[p]} \right\}_{p=1}^{P} \right), \tag{5.19}$$

where U is the number of unique patches that are present. Since $f(X) = \mathbf{f}$ is constructed as a sum of patch responses, and since all patch responses are present in $\mathbf{u}$, we can describe the deterministic and linear relationship between the two using the matrix $W \in \{0,1\}^{N \times U}$, with $W_{nm} = 1$ if the patch $\mathbf{z}_m$ is present in the image $\mathbf{x}_n$:

$$\mathbf{f} = W\mathbf{u} \,. \tag{5.20}$$

Finally, we use the (reduced) Singular Value Decomposition, giving $W = U_R S_R V_R^{\mathsf{T}}$. For $\operatorname{rank} W = \rho$, $U_R \in \mathbb{R}^{N \times \rho}$, $S_R \in \mathbb{R}^{\rho \times \rho}$ and is diagonal, and $V_R \in \mathbb{R}^{U \times \rho}$. The columns of U_R and V_R are mutually orthonormal and respectively span the column and row spaces of W.

Lemma 2. *For general, point-wise likelihoods, and priors using the convolutional kernel of equation* 5.13*, the parameters of $q(f(Z))$ may be constrained without reducing the variational lower bound. We take $\boldsymbol{\mu}_{\mathbf{f}} = U_R S_R \boldsymbol{\mu}_{\|}$ and $\Sigma_{\mathbf{f}} = U_R S_R \Sigma_{\|} S_R U_R^{\intercal}$ (with the full-rank S_R only being included for algebraic convenience). In this case the KL term in the variational lower bound (equation* 2.36*) becomes equal to*

$$\mathrm{KL}[\, q(f(Z)) \,\|\, p(f(Z)) \,] = \mathrm{KL}\Big[\mathcal{N}\Big(\boldsymbol{\alpha}_{\|}; \boldsymbol{\mu}_{\|}, \Sigma_{\|}\Big) \,\Big\|\, \mathcal{N}\Big(\boldsymbol{\alpha}_{\|}; 0, V_R^T K_{\mathbf{uu}} V_R\Big) \Big] . \tag{5.21}$$

Proof. As usual, the variational bound is $\mathcal{L} = \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[\, q(f(Z)) \,\|\, p(f(Z))\,]$. With the notation introduced above, we can write the prior as

$$p(f(Z)) = \mathcal{N}(f(Z); 0, W K_{\mathbf{uu}} W^\mathsf{T})\,. \tag{5.22}$$

If $q(f(Z))$ places density where $p(f(Z))$ does not, the KL term will be infinite. For a full-rank kernel $K_{\mathbf{uu}}$, $p(f(Z))$ places density only in the column space of W, which is spanned by the columns of U_R. We can constrain $q(f(Z))$ to lie within this space by taking $\boldsymbol{\mu}_\mathbf{f} = U_R \boldsymbol{\mu}$ and $\Sigma_\mathbf{f} = U_R \Sigma U_R^\mathsf{T}$. This only removes solutions that would otherwise have had a lower bound of $-\infty$, so this does not constitute a loss of generality.

We now can rewrite the KL divergence into the desired form by the change of variables $\mathbf{f} = U_R S_R \boldsymbol{\alpha}_\parallel$:

$$\begin{aligned}\mathrm{KL}[\, q(\mathbf{f}) \,\|\, p(\mathbf{f})\,] &= \mathrm{KL}\Big[\mathcal{N}\Big(U_R S_R \boldsymbol{\mu}_\parallel, U_R S_R \Sigma_\parallel S_R U_R^\mathsf{T}\Big) \,\Big\|\, \mathcal{N}(0, U_R S_R V_R^\mathsf{T} K_{\mathbf{uu}} V_R S_R U_R^\mathsf{T})\Big] \\ &= \mathrm{KL}\Big[\mathcal{N}\Big(\boldsymbol{\alpha}_\parallel; \boldsymbol{\mu}_\parallel, \Sigma_\parallel\Big) \,\Big\|\, \mathcal{N}\Big(\boldsymbol{\alpha}_\parallel, 0, V_R^\mathsf{T} K_{\mathbf{uu}} V_R\Big)\Big]\end{aligned} \tag{5.23}$$

where $V_R^\mathsf{T} K_{\mathbf{uu}} V_R$ is always full rank. □

The constraint on $\Sigma_\mathbf{f}$ occurs when $\operatorname{rank} W < N$, which must occur when $U < N$, i.e. the number of unique patches is smaller than the number of images, and implies a deterministic relationship between the function values for different images. In turn, this implies that we can obtain the same bound with $U < N$ inducing points, so long as the inducing points preserve the deterministic relationship. This is the case when the chosen inducing points contain all the patches that are present in the input images.

Remark. *The convolutional kernel can be recovered exactly with fewer than N inducing points, when the number of unique patches is smaller than the number of images $U < N$. The inducing points are required to contain all unique patches.*

5.4.2 Inducing patches

For inducing patches, we can make a similar argument as in lemma 1 by starting from the KL divergence between processes for approximate posteriors constructed from inter-domain variables [Matthews, 2016]. Here we denote the inducing variables associated with inducing patches as $\mathbf{u}$, and KL divergence of the resulting approximation as $D_\mathbf{u}$. The KL becomes:

$$D_\mathbf{u} = \mathrm{KL}[\, q(f) \,\|\, p(f|\mathbf{y})\,] = \mathrm{KL}[\, q(f(X), \mathbf{u}) \,\|\, p(f(X), \mathbf{u}|\mathbf{y})\,]\,. \tag{5.24}$$

This expression is harder to justify from the reasoning presented in section 2.4.2, as it contains variables that are not directly part of the GP, and proved using a measure theoretic argument that depends on our inducing variables $\mathbf{u}$ being deterministically related to the process of interest. However, the expression above allows us to analyse the KL between processes without measure theory, and prove a result for inducing patches that is similar to lemma 1.

Lemma 3. *For general, point-wise likelihoods $p(y_n|f(\mathbf{x}_n))$, and the convolutional kernel from equation* 5.13*, the optimal inducing variables defined by inducing patches $\mathbf{u} = g(Z)$ are obtained by using the collection of all unique patches in all training points as the inducing patches $Z = \bigcup_{n=1}^{N} \left\{\mathbf{x}_n^{[p]}\right\}_{p=1}^{P}$.*

Proof. As before, we start from the KL between processes, with an extra inducing variable added, here denoted as $\mathbf{u}^*$:

$$\begin{aligned} D_{\mathbf{u}} &= \mathrm{KL}[\, q(f(X), \mathbf{u}, \mathbf{u}^*) \,\|\, p(f(X), \mathbf{u}, \mathbf{u}^*|\mathbf{y})\,]\,, \\ &= \mathrm{KL}[\, p(f(X)|\mathbf{u}, \mathbf{u}^*) q(\mathbf{u}, \mathbf{u}^*) \,\|\, p(f(X)|\mathbf{u}, \mathbf{u}^*, \mathbf{y}) p(\mathbf{u}, \mathbf{u}^*|\mathbf{y}))\,]\,. \end{aligned} \tag{5.25}$$

From the construction of $f(\cdot)$ in equation 5.12, and the definition of Z, we see that there is a deterministic relationship between $f(X)$ and $\mathbf{u}$. Given this deterministic relationship, the conditionals $p(f(X)|\mathbf{u}, \mathbf{u}^*, \mathbf{y})$ and $p(f(X)|\mathbf{u}, \mathbf{u}^*)$ become independent of $\mathbf{u}^*$ and $\mathbf{y}$, and are therefore equal on both sides of the KL, allowing them to be removed from consideration:

$$D_{\mathbf{u}} = \mathrm{KL}[\, \cancel{p(f(X)|\mathbf{u}, \mathbf{u}^*)} q(\mathbf{u}, \mathbf{u}^*) \,\|\, \cancel{p(f(X)|\mathbf{u}, \mathbf{u}^*)} p(\mathbf{u}, \mathbf{u}^*|\mathbf{y}))\,]\,. \tag{5.26}$$

The KL is now in the same situation as in lemma 1, where we have the optimal KL, and a positive term:

$$\mathrm{KL} = \underbrace{\mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u}|\mathbf{y})\,]}_{\text{KL without additional } \mathbf{u}^*} + \underbrace{\mathbb{E}_{q(\mathbf{u})}[\mathrm{KL}[\, q(\mathbf{u}^*|\mathbf{u}) \,\|\, p(\mathbf{u}^*|\mathbf{u})\,]]}_{\geq 0}\,, \tag{5.27}$$

which implies that the KL can only be increased by adding additional inducing variables. □

5.4.3 Comparing inducing points and inducing patches

Lemmas 1 and 3 tell us the optimal inducing inputs for both approximations. Here we will also consider the influence of the respective Gaussian $q(\cdot)$ distributions over the inducing outputs, and show that the optimal posteriors are equivalent.

Theorem 1. *For the convolutional kernel, the optimal approximate posteriors based on inducing points and inducing patches both have the same KL divergence to the true posterior, i.e.:*

$$\min_{\boldsymbol{\mu}_\mathbf{u}, \Sigma_\mathbf{u}} \mathrm{KL}\Big[\, q_{\boldsymbol{\mu}_\mathbf{u}, \Sigma_\mathbf{u}}(\mathbf{u})) \,\Big\|\, p(\mathbf{u}|\mathbf{y}) \,\Big] = \min_{\boldsymbol{\mu}_\mathbf{f}, \Sigma_\mathbf{f}} \mathrm{KL}\Big[\, q_{\boldsymbol{\mu}_\mathbf{f}, \Sigma_\mathbf{f}}(f(X)) \,\Big\|\, p(f(X)|\mathbf{y}) \,\Big]\,. \tag{5.28}$$

Proof. To prove the statement, we will show that the variational marginal likelihood lower bounds for both approximations are equal when certain degrees of freedom of their respective parameterisations are optimised out. We already did this for inducing points in lemma 2. For the compressed parameterisation, we find the bound for the inducing points approximation by substituting the equivalent KL divergence, and the other deterministic relationships:

$$\mathcal{L}_\mathbf{f} = \mathbb{E}_{\mathcal{N}\left(\boldsymbol{\alpha}_\parallel; \boldsymbol{\mu}_\parallel, \Sigma_\parallel\right)}\Big[\log p(\mathbf{y}|U_R S_R \boldsymbol{\alpha}_\parallel)\Big] - \mathrm{KL}\Big[\, \mathcal{N}\Big(\boldsymbol{\alpha}_\parallel; \boldsymbol{\mu}_\parallel, \Sigma_\parallel\Big) \,\Big\|\, \mathcal{N}\Big(\boldsymbol{\alpha}_\parallel; 0, V_R^\intercal K_{\mathbf{uu}} V_R\Big) \,\Big] \tag{5.29}$$

The inducing patch bound can be written in exactly the same way. We start with the standard bound, and then substitute relationship $\mathbf{f} = W\mathbf{u}$:

$$\mathcal{L}_\mathbf{u} = \mathbb{E}_{q(\mathbf{u})}[\log p(\mathbf{y}|W\mathbf{u})] - \mathrm{KL}[\, \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_\mathbf{u}, \Sigma_\mathbf{u}) \,\|\, \mathcal{N}(\mathbf{u}; 0, K_{\mathbf{uu}}) \,]\,. \tag{5.30}$$

If W has a space for which $W\mathbf{u} = 0$, certain degrees of freedom in $\Sigma_\mathbf{u}$ are unconstrained by the likelihood. We can then strictly improve the bound by minimising the KL term with respect to these degrees of freedom. A full SVD on W gives us

$$W = USV^\intercal = \begin{bmatrix} U_R & U_N \end{bmatrix} S \begin{bmatrix} V_R^\intercal \\ V_N^\intercal \end{bmatrix} \tag{5.31}$$

where the columns of V_N gives us an orthonormal basis for the null space of W. Since V is invertible and orthonormal, we can write $\boldsymbol{\alpha} = V^\intercal \mathbf{u}$ and $\mathbf{u} = V\boldsymbol{\alpha} = V_R\boldsymbol{\alpha}_\parallel + V_N\boldsymbol{\alpha}_\perp$. $\boldsymbol{\alpha}_\perp$ are the coefficients for the components of $\mathbf{u}$ that lie within the nullspace of W, and

therefore do not affect the expected likelihood term. If we parameterise $q(\mathbf{u})$ as

$$\boldsymbol{\mu}_{\mathbf{u}} = V_R \boldsymbol{\mu}_{\|} + V_N \boldsymbol{\mu}_{\perp} \,, \tag{5.32}$$

$$\Sigma_{\mathbf{u}} = V \Sigma_{\boldsymbol{\alpha}} V^{\intercal} = V_R \Sigma_{\|} V_R^{\intercal} + V_R \Sigma_{12} V_N^{\intercal} + V_N \Sigma_{21} V_R^{\intercal} + V_N \Sigma_{\perp} V_N^{\intercal} \,, \tag{5.33}$$

then we are free to choose $\boldsymbol{\mu}_{\perp}$, Σ_{12}, and $\Sigma_{\perp}$ while only affecting the KL term. We now find the optimal values for these variables, starting by rewriting the KL divergence of $\mathcal{L}_{\mathbf{u}}$ in terms of $\boldsymbol{\alpha}$:

$$\begin{aligned} &\mathrm{KL}[\,\mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_{\mathbf{u}}, \Sigma_{\mathbf{u}}) \,\|\, \mathcal{N}(\mathbf{u}; 0, K_{\mathbf{uu}})\,] = \mathrm{KL}[\,\mathcal{N}(\boldsymbol{\alpha}; \boldsymbol{\mu}_{\boldsymbol{\alpha}}, \Sigma_{\boldsymbol{\alpha}}) \,\|\, \mathcal{N}(\boldsymbol{\alpha}; 0, V^{\intercal} K_{\mathbf{uu}} V)\,] \\ &\quad = \mathrm{KL}\left[\mathcal{N}\left(\begin{bmatrix} \boldsymbol{\alpha}_{\|} \\ \boldsymbol{\alpha}_{\perp} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_{\|} \\ \boldsymbol{\mu}_{\perp} \end{bmatrix}, \begin{bmatrix} \Sigma_{\|} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{\perp} \end{bmatrix} \right) \middle\| \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\alpha}_{\|} \\ \boldsymbol{\alpha}_{\perp} \end{bmatrix}; 0, \begin{bmatrix} V_R^{\intercal} K_{\mathbf{uu}} V_R & V_R^{\intercal} K_{\mathbf{uu}} V_N \\ V_N^{\intercal} K_{\mathbf{uu}} V_R & V_N^{\intercal} K_{\mathbf{uu}} V_N \end{bmatrix} \right) \right] . \end{aligned} \tag{5.34}$$

We can factorise the KLs in order to separate out $\boldsymbol{\alpha}_{\perp}$, which describes the component of $\mathbf{u}$ in the null space of W.

$$\mathrm{KL}[\, q(\boldsymbol{\alpha}) \,\|\, p(\boldsymbol{\alpha})\,] = \mathrm{KL}\Big[\, q(\boldsymbol{\alpha}_{\|}) \,\Big\|\, p(\boldsymbol{\alpha}_{\|}) \,\Big] + \mathbb{E}_{q(\boldsymbol{\alpha}_{\|})}\Big[\mathrm{KL}\Big[\, q(\boldsymbol{\alpha}_{\perp}|\boldsymbol{\alpha}_{\|}) \,\Big\|\, p(\boldsymbol{\alpha}_{\perp}|\boldsymbol{\alpha}_{\|}) \,\Big]\Big] \tag{5.35}$$

The conditional distributions are given by:

$$p(\boldsymbol{\alpha}_{\perp}|\boldsymbol{\alpha}_{\|}) = \mathcal{N}\Big(\boldsymbol{\alpha}_{\perp}; V_N^{\intercal} K_{\mathbf{uu}} V_R (V_N K_{\mathbf{uu}} V_R)^{-1} \boldsymbol{\alpha}_{\|}, \tag{5.36}$$

$$V_N^{\intercal} K_{\mathbf{uu}} V_N - V_N^{\intercal} K_{\mathbf{uu}} V_R (V_R^{\intercal} K_{\mathbf{uu}} V_R)^{-1} V_R^{\intercal} K_{\mathbf{uu}} V_N \Big) \tag{5.37}$$

$$q(\boldsymbol{\alpha}_{\perp}|\boldsymbol{\alpha}_{\|}) = \mathcal{N}\Big(\boldsymbol{\alpha}_{\perp}; \boldsymbol{\mu}_{\perp} + \Sigma_{21} \Sigma_{\|}^{-1} (\boldsymbol{\alpha}_{\|} - \boldsymbol{\mu}_{\|}), \Sigma_{\perp} - \Sigma_{21} \Sigma_{\|}^{-1} \Sigma_{12} \Big) \tag{5.38}$$

We can reduce the conditional KL to 0 by choosing

$$\Sigma_{21} = V_N^{\intercal} K_{\mathbf{uu}} V_R (V_R^{\intercal} K_{\mathbf{uu}} V_R)^{-1} \Sigma_{\|} \,, \tag{5.39}$$

$$\boldsymbol{\mu}_{\perp} = \Sigma_{21} \Sigma_{11}^{-1} \boldsymbol{\mu}_{\|} \,, \tag{5.40}$$

$$\Sigma_{\perp} = V_N^{\intercal} K_{\mathbf{uu}} V_N - V_N^{\intercal} K_{\mathbf{uu}} V_R (V_R^{\intercal} K_{\mathbf{uu}} V_R)^{-1} V_R^{\intercal} K_{\mathbf{uu}} V_N + \Sigma_{21} \Sigma_{\|}^{-1} \Sigma_{21} \,. \tag{5.41}$$

We can now rewrite the inducing patch bound as

$$\mathcal{L}_{\mathbf{u}} = \mathbb{E}_{q(\boldsymbol{\alpha}_{\|})}\Big[\log p(\mathbf{y}|U_R S_R \boldsymbol{\alpha}_{\|}) \Big] - \mathrm{KL}\Big[\, \mathcal{N}\Big(\boldsymbol{\alpha}_{\|}; \boldsymbol{\mu}_{\|}, \Sigma_{\|} \Big) \,\Big\|\, \mathcal{N}\Big(\boldsymbol{\alpha}_{\|}; 0, V_R^{\intercal} K_{\mathbf{uu}} V_R \Big) \,\Big] \tag{5.42}$$

This is exactly the same form as the form for $\mathcal{L}_{\mathbf{f}}$ in equation 5.29. We have now shown that for particular restrictions of $\boldsymbol{\mu}_{\mathbf{u}}$, $\Sigma_{\mathbf{u}}$, $\boldsymbol{\mu}_{\mathbf{f}}$, and $\Sigma_{\mathbf{f}}$ that contain the optimal

parameters, the bounds $\mathcal{L}_{\mathbf{u}}$ and $\mathcal{L}_{\mathbf{f}}$ are the same. This implies that the optimal variational distributions give the same KL divergence to the posterior. □

Remark. *The optimal inducing patch approximation is equivalent to the optimal inducing point approximation, and only needs more inducing variables if the number of unique patches in all input images is larger than the number of images.*

This result shows that the inducing patch approximation is as compact as the most compact inducing point approximation, when considering optimal inducing placement of the inducing inputs. It is harder to make precise statements when the number of inducing variables is limited more strongly, for both inducing point and patch approximations. However, the same intuition holds: if the inducing variables constrain the function well for the training data, the approximation is likely to be good. For inducing patches, this implies that if the variation in terms of the present patches is small, there is likely to be a compact inducing patch approximation. We now move on to investigating several variants of the convolutional kernel in practice.

5.5 Translation invariant convolutional kernels

We call the basic version of the convolutional kernel as described in section 5.3.1 "translation invariant", as the influence of a particular patch on the overall function f will be equal regardless of its location. We re-state the inter-domain covariances for completeness:

$$k_{fu}(\mathbf{x}, \mathbf{z}) = \mathbb{E}_g\left[\sum_p g(\mathbf{x}^{[p]})g(\mathbf{z})\right] = \sum_p k_g\Big(\mathbf{x}^{[p]}, \mathbf{z}\Big)\,, \tag{5.43}$$

$$k_{uu}(\mathbf{z}, \mathbf{z}') = k_g(\mathbf{z}, \mathbf{z}')\,. \tag{5.44}$$

In order to highlight the capabilities of the new kernel and the accompanying approximation, we now consider two toy tasks: classifying rectangles and distinguishing zeros from ones in MNIST.

5.5.1 Toy demo: rectangles

The rectangles dataset is an artificial dataset containing 1200 images of size 28×28. Each image contains the outline of a randomly generated rectangle, and is labelled according to whether the rectangle has larger width or length. Despite its simplicity, the dataset is tricky for common kernels that rely on Euclidean distances, because

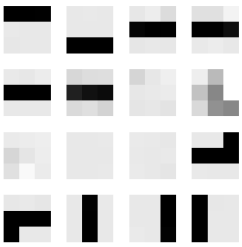


Fig. 5.6 The optimised inducing patches on the rectangles dataset.

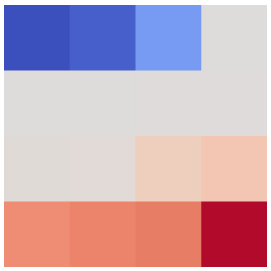


Fig. 5.7 The corresponding function of $\mu_{\mathbf{u}}$. Blue is low, red is high.

of the high dimensionality of the input, and the strong dependence of the label on multiple pixel locations.

To tackle the rectangles dataset with the convolutional GP, we used a patch-size of 3×3. We set the number of inducing points to 16, and initialised them with uniform random noise. We optimised using Adam [Kingma and Ba, 2014], using a learning rate of 0.01 with 100 points per minibatch, and obtained 1.4% error and a negative log predictive probability (nlpp) of 0.055 on the held out test set. For comparison, a squared exponential kernel with 1200 optimally placed inducing points, optimised with BFGS gives 5.0% error and an nlpp of 0.258. The convolutional model performs better, with fewer inducing points. Note that for the squared exponential model variational inference is only used to handle the non-conjugate likelihood, rather than to obtain a sparse inducing point posterior.

The model works because it is able to recognise and count vertical and horizontal bars in the patches. We can see this by considering the inducing patches (figure 5.6) and their corresponding outputs (figure 5.7). The inducing patches are sorted by the mean value of their corresponding inducing output. Horizontal bars have negative values, while vertical bars are positive. Each patch contributes a value to the output of the GP over the entire image, and the classification will depend on whether the majority is horizontal or vertical. Perfect classification is prevented by edge effects. In the centre of the image, a vertical bar contributes to the output of $f(\cdot)$ in 3 individual patches. However, at the edge, a vertical bar will be captured only once.

5.5.2 Illustration: Zeros vs ones MNIST

We perform a similar experiment for classifying MNIST 0 and 1 digits for illustrative purposes. We use the same experimental setup with 50 inducing features, shown in figure 5.8. The inducing patches are sorted by the mean of their inducing output,

ranging from -1.51 in the top left to $+1.52$ in the bottom right. This gives insight into the types of features the GP takes as evidence for a zero or one being present, negative in favour for zero and positive in favour of one. Features for zeros tend to be diagonal or bent lines, while features for ones tend to be blank space or thick edges. We obtain 99.7% accuracy.

Fig. 5.8 The optimised inducing patches on the MNIST 0-vs-1 dataset with the translation invariant convolutional kernel.

5.5.3 Full MNIST

Next, we turn to the full multi-class MNIST dataset. Our model uses 10 independent latent Gaussians using the convolutional kernel with 5×5 patches. We use the same setup as in Hensman et al. [2015a], only constraining $q(\mathbf{u})$ to a Gaussian. It seems that the translation invariant kernel is too restrictive for this task, since its accuracy plateaus at around 95.7%, compared to 97.9% for the squared exponential kernel (figure 5.11). This is consistent with the poor marginal likelihood bound (figure 5.10). It appears that the summation structure here is not flexible enough. Full results are in table 5.1.

5.6 Weighted convolutional kernels

We saw in the previous section that although the fully translation invariant kernel excelled at the rectangles task, it under-performed compared to the squared exponential on MNIST. Full translation invariance is too strong a constraint, which makes intuitive sense for image classification, as the same feature in different locations of the image can imply different classes. This can be easily avoided, without leaving the family of Gaussian processes, by weighting the response from each patch. Denoting again the underlying patch-based GP as $g(\cdot)$, the image-based GP $f(\cdot)$ is given by

$$f(\mathbf{x}) = \sum_p w_p g(\mathbf{x}^{[p]}) \,. \tag{5.45}$$

The weights w_p can be inferred in order to adjust the relative importance of the response for each location in the image.

In order to use this model construction in the variational GP framework we require again the covariance function k_f as well as the inter-domain covariance function k_{fg}:

$$k_f(\mathbf{x}, \mathbf{x}) = \sum_{pq} w_p w_q k_g(\mathbf{x}^{[p]}, \mathbf{x}_q) \,, \tag{5.46}$$

$$k_{fg}(\mathbf{x}, \mathbf{z}) = \sum_p w_p k_g(\mathbf{x}^{[p]}, \mathbf{z}) \,. \tag{5.47}$$

The patch weights $\mathbf{w} \in \mathbb{R}^P$ are considered to be kernel hyperparameters: we optimise them with respect to the marginal likelihood lower bound in the same fashion as the underlying parameters of the kernel k_g. This introduces P hyperparameters into the kernel, which is slightly less than the number of input pixels, which is how many hyperparameters a squared exponential kernel with automatic relevance determination would have. This may lead to slight over-fitting and over-estimation of the marginal likelihood. Inference for $\mathbf{w}$ could be extended to MAP, MCMC or variational methods, if overfitting presents a large problem.

5.6.1 Toy demo: Rectangles

The errors in the previous section were caused by rectangles along the edge of the image, which contained bars which only contribute once to the classification score. Bars in the centre contribute to multiple patches. The weighting allows some up-weighting of patches along the edge. This results in near-perfect classification, with no classification errors and an nlpp of 0.005 in the test set.

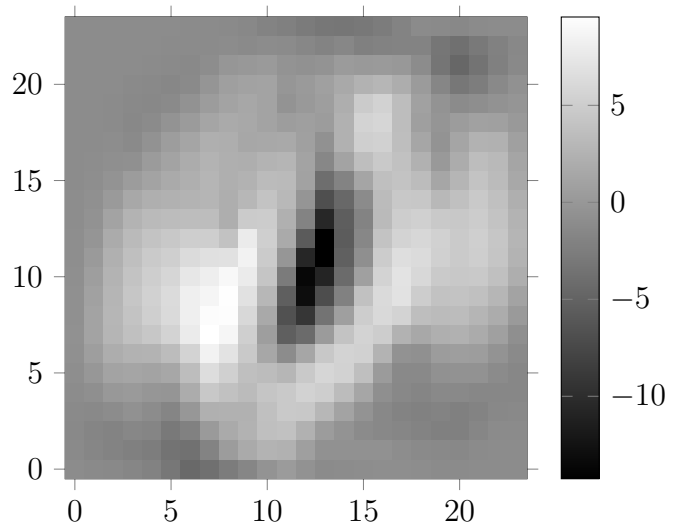


Fig. 5.9 The patch response kernel hyperparameters for the weighted convolutional kernel on the MNIST 0-vs-1 dataset.

5.6.2 Illustration: Zeros vs ones MNIST

Using the same experimental setup as earlier, we repeat the classification of MNIST 0 and 1 digits for illustration purposes. Due to the weighting of a patch's response, we can not interpret $\boldsymbol{\mu}_{\mathbf{u}}$ in the same way as earlier, where the value indicated the strength of evidence towards a particular class. Depending on the location of a patch, the same patch could contribute towards a positive *or* negative classification. However, the weights do give insight into what regions of the image carry importance for either class. Figure 5.9 shows the patch weights, which nicely highlight exactly the difference in areas where salient features for ones and zeros would be. This modification obtained perfect classification on the test set.

5.6.3 Full MNIST

By adding the weighting, we see a significant reduction in error over the translation invariant and squared exponential kernels (table 5.1 & figure 5.11). The weighted convolution kernel obtains 1.38% error – a significant improvement over 1.95% for the squared exponential kernel [Hensman et al., 2015a]. Krauth et al. [2016] report 1.55% error using a squared exponential kernel, but using a Leave-One-Out objective for finding the hyperparameters. The variational lower bound correctly identifies the best performing model, if it were to be used for model comparison (figure 5.10).

5.7 Is convolution too much invariance?

As discussed earlier, the additive nature of the convolution kernel places constraints on the possible functions in the prior. While these constraints have been shown to be useful for classifying MNIST, the non-parametric nature of the squared exponential, which guarantees enough capacity to model well in the large data limit, is lost: convolutional kernels are not universal [Rasmussen and Williams, 2005, §4.3] in the image input space. Here we investigate whether a small squared exponential component in the kernel can push classification performance even further.

5.7.1 Convolutional kernels are not universal

Kernels with invariances and convolutional kernels are odd in the sense that they are non-parametric, but not universal. They are non-parametric, as the Mercer expansion (equation 1.29) has an infinite number of terms, however, they lack the ability to approximate any function arbitrarily closely. For invariant kernels this fact is obvious: the function is constrained to have the same value for different inputs. Any function that does not obey this constraint, cannot be captured. Weighted convolutional kernels, on the other hand, are slightly more complicated since they have given up their exact invariance to patch permutations. To show that they are not universal, we need to construct a set of inputs which would fully constrain the function value at a different input. This would show up as a rank deficient covariance matrix.

Claim. *Weighted covariance kernels are not universal.*

Proof. Consider N distinct images with P (not necessarily distinct) patches each. An example of this would be a 28×28 image with only a single pixel on in different locations. This would give 784 possible images, that could have possible different responses. Our N input images contain U distinct patches. Assuming 5×5 patches, $U = 25$. We collect the response for each individual patch in $\mathbf{g}$. The function values $\mathbf{f}$ can be obtained through the linear transformation

$$\mathbf{f} = WQ\mathbf{g}\,, \tag{5.48}$$

where W simply has the patch weights as rows, and Q projects $\mathbf{g}$ to a vector of patch responses at the correct locations. The product WQ has size $N \times U$. This implies that the convolutional function evaluated at the $U + 1$th image in the example above has to be fully determined by the responses of the previous images. This image exists, as $N > U$.

Of course, the kernel matrix for these inputs has to have some zero eigenvalues because the matrix

$$\mathbb{E}[\mathbf{f}\mathbf{f}^\intercal] = \mathbb{E}[WG\mathbf{g}\mathbf{g}^\intercal W^\intercal G^\intercal] = WQK_G Q^\intercal W^\intercal \tag{5.49}$$

has rank at most U. □

This simple example shows what functions on images the convolutional kernel can not model. It is possible that the classification performance we obtained is held back by the constrained nature of the convolutional kernel, despite the very same constraints being necessary to outperform the squared exponential kernel on its own. In the next section, we attempt to get the best of both worlds of constrained convolutional kernels and flexible squared exponentials.

5.7.2 Adding a characteristic kernel component

We can again obtain a kernel on $f(\cdot)$ that is both characteristic and has convolutional structure by adding a characteristic squared exponential (SE) component with a convolutional component: $f(\mathbf{x}) = f_{conv}(\mathbf{x}) + f_{rbf}(\mathbf{x})$. This construction allows any residuals that the convolutional structure cannot explain to be modelled by the characteristic SE kernel. By adjusting the variance parameters of both kernels, we can interpolate between the behaviour of the two kernels on their own. Having a too small a variance for the characteristic component will not allow enough of the residuals of the convolutional component to be modelled, while having too large a variance will result in a model that is too flexible to generalise again. Following section 5.1.1, we use the marginal likelihood (or rather, an approximation to it) to weigh these competing considerations and choose the variance parameters, in the same way as was done in other additive models [Duvenaud et al., 2013, 2011].

Inference in such a model is straightforward under the normal inducing point framework – it requires only plugging in the resulting sum of kernels. Our case is complicated since we can not place the inducing inputs for the SE component in patch space. We need the inducing inputs for the SE to lie in the space of images, while we want to use inducing patches for the convolutional kernel. This forces us to use a slightly different form for the approximating GP, representing the inducing inputs and

outputs for each component separately, as

$$\begin{bmatrix}\mathbf{u}_{conv}\\ \mathbf{u}_{rbf}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{conv}\\ \boldsymbol{\mu}_{rbf}\end{bmatrix}, \mathbf{S}\right), \tag{5.50}$$

$$f(\cdot)\,|\,\mathbf{u} = f_{conv}(\cdot)\,|\,\mathbf{u}_{conv} + f_{rbf}(\cdot)\,|\,\mathbf{u}_{rbf}\,. \tag{5.51}$$

We can derive a similar variational lower bound as in equation 2.36. For clarity, we are interested in a model where we have a prior which is the sum of two GPs, and an arbitrary likelihood:

$$f_1(\cdot) \sim \mathcal{GP}(0, k_1(\cdot,\cdot))\,, \tag{5.52}$$

$$f_2(\cdot) \sim \mathcal{GP}(0, k_2(\cdot,\cdot))\,, \tag{5.53}$$

$$f(\cdot) = f_1(\cdot) + f_2(\cdot)\,, \tag{5.54}$$

$$y_i|f, \mathbf{x}_i \sim p(y_i|f(\mathbf{x}_i))\,. \tag{5.55}$$

We can write down the probability of everything and introduce a bound using the trick in Titsias [2009b] of matching the prior conditionals. The component GPs $f_1(\cdot)$ and $f_2(\cdot)$ are independent in the prior. The inducing variables $\mathbf{u}_n$ are observations of the component GPs, possibly in different domains.

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y}|f(\mathbf{x}))p(f(\mathbf{x})|\mathbf{u}_1,\mathbf{u}_2)p(\mathbf{u}_1)p(\mathbf{u}_2)\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \tag{5.56}$$

$$\geq \log \int \frac{p(\mathbf{y}|f(\mathbf{x}))\cancel{p(f(\mathbf{x})|\mathbf{u}_1,\mathbf{u}_2)}p(\mathbf{u}_1)p(\mathbf{u}_2)}{\cancel{p(f(\mathbf{x})|\mathbf{u}_1,\mathbf{u}_2)}q(\mathbf{u}_1,\mathbf{u}_2)}\mathrm{d}\mathbf{f}_1\mathrm{d}\mathbf{f}_2\mathrm{d}\mathbf{u} \tag{5.57}$$

$$= \sum_n \mathbb{E}_{q(f(\mathbf{x}_n))}[p(y_n|f(\mathbf{x}_n))] - \mathrm{KL}[\,q(\mathbf{u}_1,\mathbf{u}_2)\,\|\,p(\mathbf{u}_1)p(\mathbf{u}_2)\,] \tag{5.58}$$

The marginal distributions needed for the likelihood expectation can simply be found by considering the conditional of $f(\cdot)$ and remembering that the sum of two independent Gaussian processes is a Gaussian process with the means and covariances summed:

$$f(\cdot)|\mathbf{u}_1,\mathbf{u}_2 \sim \mathcal{GP}\left(\sum_i \mathbf{k}_{\mathbf{u}_i}^\mathsf{T}(\cdot)K_{\mathbf{u}_i\mathbf{u}_i}^{-1}\mathbf{u}_i, \sum_i k_i(\cdot,\cdot) - \mathbf{k}_{\mathbf{u}_i}^\mathsf{T}(\cdot)K_{\mathbf{u}_i\mathbf{u}_i}^{-1}\mathbf{k}_{\mathbf{u}_i}(\cdot)\right). \tag{5.59}$$

We can choose a factorised approximation between $\mathbf{u}_1$ and $\mathbf{u}_2$, or a full joint, where we learn a $2M \times 2M$ posterior covariance matrix. In the former case, the predictive variance of $q(f(\mathbf{x}_n))$ is simply the sum of predictive covariances of two sparse GP posteriors, while in the latter case some extra interaction terms are required. By denoting $\mathbf{S}_{ii'}$ as the approximate posterior covariance between $\mathbf{u}_i$ and $\mathbf{u}_{i'}$, we obtain

the predictive variance for the non-factorised version

$$\sigma_n^2 = k(\mathbf{x}_n, \mathbf{x}_n) + \sum_i \sum_{i'} \delta_{ii'} \mathbf{k}_{\mathbf{u}_i}(\mathbf{x}_n)^\intercal K^{-1}_{\mathbf{u}_i\mathbf{u}_i} \mathbf{S}_{ii'} K^{-1}_{\mathbf{u}_{i'}\mathbf{u}_{i'}} \mathbf{k}_{\mathbf{u}'_i}(\mathbf{x}_n) - \mathbf{k}_{\mathbf{u}_i}(\mathbf{x}_n) K^{-1}_{\mathbf{u}_i\mathbf{u}_i} \mathbf{k}_{\mathbf{u}_i}(\mathbf{x}_n)\,, \tag{5.60}$$

in which we never need expensive matrix operations on matrices larger than $M \times M$, despite dealing with $2M$ inducing variables. A similar simplification can be done in the KL divergence.

5.7.3 MNIST

By adding a squared exponential component, we indeed get an extra reduction in error from 1.38% to 1.17% (table 5.1 & figure 5.11). The variances for the convolutional and SE kernels are $1.13 \cdot 10^{-2}$ and $3.82 \cdot 10^{-4}$ respectively. The marginal likelihood lower bound of the convolutional + SE model is slightly higher than that of the weighted convolutional model.

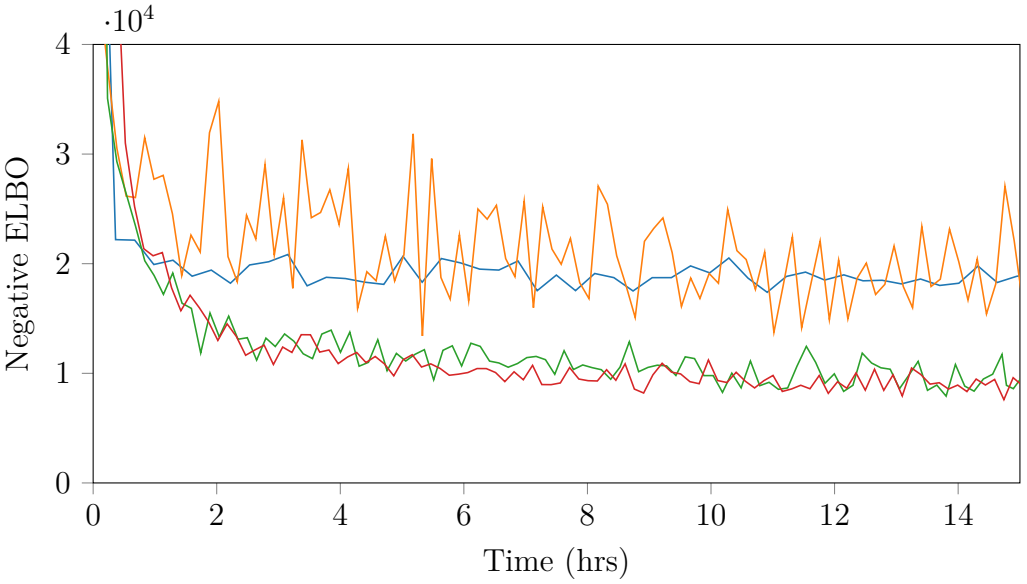


Fig. 5.10 Optimisation of the variational lower bound for the full MNIST experiment for 4 different kernels: SE (blue), translation invariant convolutional (orange), weighted convolutional (green) and weighted convolutional + SE (red) kernels.

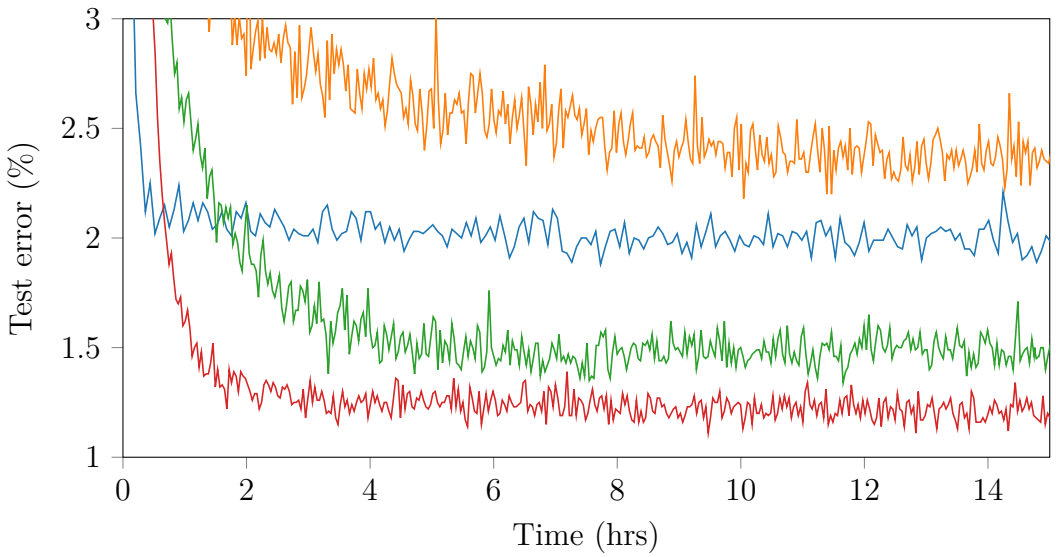


Fig. 5.11 Classification accuracies on 10 class MNIST for 4 different kernels: SE (blue), translation invariant convolutional (orange), weighted convolutional (green) and weighted convolutional + SE (red) kernels.

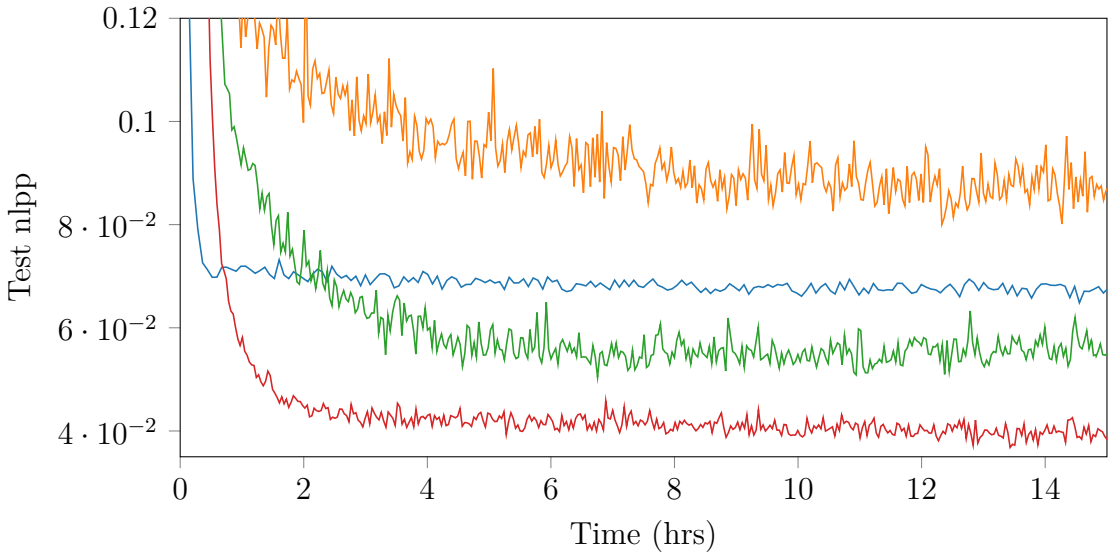


Fig. 5.12 Negative log predictive probabilities on 10 class MNIST for 4 different kernels: SE (blue), translation invariant convolutional (orange), weighted convolutional (green) and weighted convolutional + SE (red) kernels.

Kernel	M	Error (%)	NLPP
Invariant	750	2.35%	0.089
SE	750	1.95%	0.068
Weighted	750	1.38%	0.056
Weighted + SE	750	1.17%	0.039

Table 5.1 Final results for MNIST.

5.8 Convolutional kernels for colour images

Our final variants of the convolutional kernel are focused on handling images with multiple colour channels. The addition of colour channels presents an interesting modelling challenge. While there is definitely information in colour, the cost is a significant increase in input dimensionality, with a large amount of redundancy between colour channels.

As a baseline, the weighted convolutional kernel from section 5.6 can be used with the collection of patches simply containing all patches from each colour channel separately (**baseline weighted convolutional kernel**). For images with a width W, height H and number of colour channels C, we would get $P = (W - w + 1) \times (H - h + 1) \times C$ patches. This kernel can only account for linear interactions between colour channels, through the weights, and is also constrained to give the same patch response for inputs, regardless of the colour channel. A next step up would be to define the patch response function to take a $w \times h \times C$ patch with all C colour channels (**weighted colour-convolution kernel**). This trades off increasing the dimensionality of the patch response with allowing it to learn non-linear interactions between the colour channels.

As a final model, we propose to use a different patch-response function $g_c(\cdot)$ for each colour channel, weighted together linearly based on patch location and colour channel. We will refer to this approximation as the **multi-channel convolutional kernel**:

$$f(\mathbf{x}) = \sum_{p=1}^{P} \sum_{c=1}^{C} w_{pc} g_c\left(\mathbf{x}^{[pc]}\right). \tag{5.61}$$

Following the earlier approximation scheme of placing the inducing points in the patch space, we can consider the C patch response functions together as a multi-output GP $g : \mathbf{z} \to \mathbb{R}^C$. Interestingly, this leads to having multi-dimensional inducing outputs in $\mathbb{R}^C$. As before with inter-domain approximations, we only need to find the (cross-)

covariances between f and g. The main difference here is that we have a C valued vector of covariances for each inducing input (denoting $\mathbf{x}^{[pc]}$ as the pth patch of the cth colour channel):

$$k_{fg_c}(\mathbf{x}, \mathbf{z}) = \mathbb{E}_{\{g_c\}_{c=1}^{C}}\left[\sum_{pc} w_{pc} g_c\left(\mathbf{x}^{[pc]}\right) g_c(\mathbf{z})\right] = \sum_{pc} w_{pc} k_g(\mathbf{x}^{[pc]}, \mathbf{z})\,. \tag{5.62}$$

To avoid any tensor arithmetic, the vectors are stacked to give an overall $K_{\mathbf{fu}} \in \mathbb{R}^{N \times CM}$ matrix. Similarly, the covariance matrix between all inducing variables $K_{\mathbf{uu}}$ can be represented as $CM \times CM$ matrix. Thanks to the independence assumption in the prior of the $g_c(\cdot)$s, it will be block-diagonal in structure. As in section 5.7, we have the choice to represent a full $CM \times CM$ covariance matrix, or go for a mean-field approximation, requiring only C $M \times M$ matrices.

The multi-output inducing variables can be avoided if the weighting of each of the channels is constant w.r.t. the patch, i.e. $w_{pc} = w_p w_c$. In this case, the pixel response will be a weighted sum of GPs, which is itself a GP with the kernels summed:

$$f(\mathbf{x}) = \sum_p w_p \sum_c w_c g_c(\mathbf{x}^{[pc]}) = \sum_p w_p \tilde{g}(\mathbf{x}^{[pc]})\,, \tag{5.63}$$

$$\tilde{g}(\cdot) \sim \mathcal{GP}\left(0, \sum_c w_c k_c(\cdot, \cdot)\right). \tag{5.64}$$

We will refer to this as the **additive colour-convolutional kernel**, as it is equivalent to the colour-convolutional kernel with an additive kernel for the patch-response function.

5.8.1 CIFAR-10

We conclude the experiments by an investigation of CIFAR-10 [Krizhevsky et al., 2009], where 32×32 sized RGB images are to be classified. We use a similar setup to the previous MNIST experiments, by using 5×5 patches. Again, all latent functions share the same kernel for the prior, including the patch weights. We compare a regular SE kernel, the "baseline" weighted convolutional kernel, a weighted colour-convolutional kernel, an additive colour-convolutional kernel and finally a multi-channel convolutional. All models use 1000 inducing inputs.

Test errors, NLPPs and ELBOs during training are shown in figures 5.13, 5.14 and 5.15. Any convolutional structure significantly improves classification performance, however, colour interactions are particularly important. All three variants of kernels

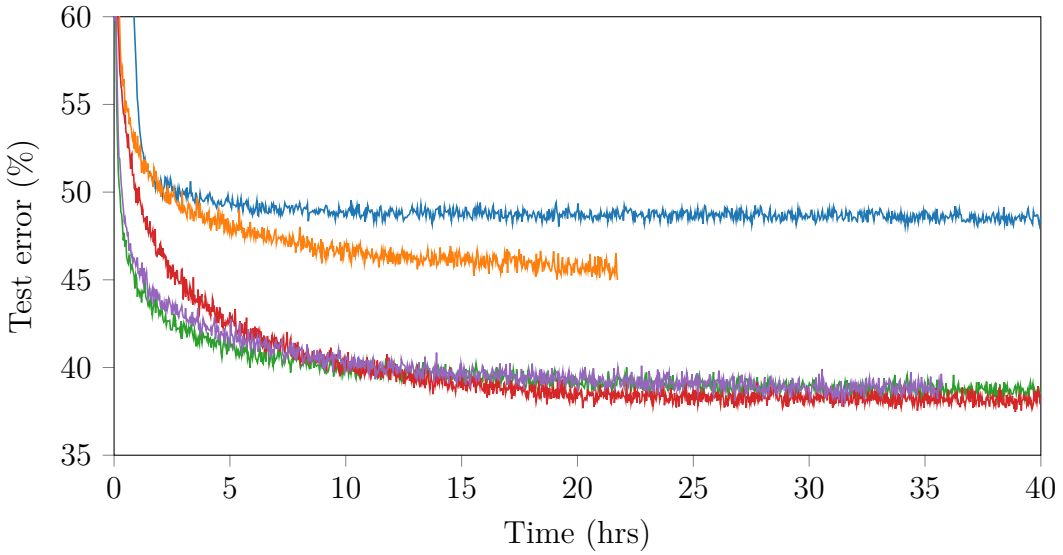


Fig. 5.13 Test error for CIFAR-10, using SE (blue), baseline weighted convolutional (orange), weighted colour-convolutional (green), additive colour-convolutional (purple) and multi-channel convolutional (red) kernels.

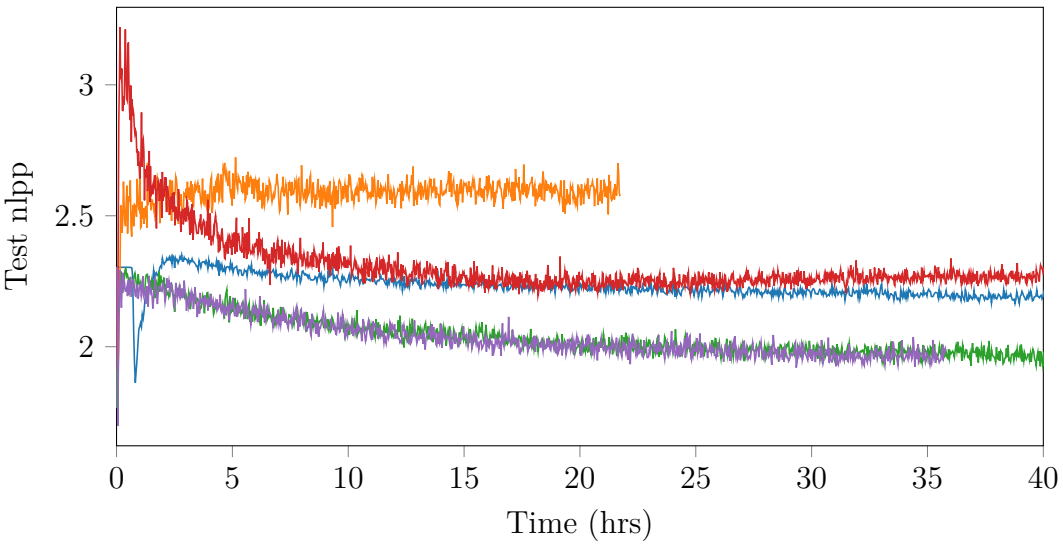


Fig. 5.14 Test negative log predictive probability (nlpp) for CIFAR-10, using SE (blue), baseline weighted convolutional (orange), weighted colour-convolutional (green), additive colour-convolutional (purple) and multi-channel convolutional (red) kernels.

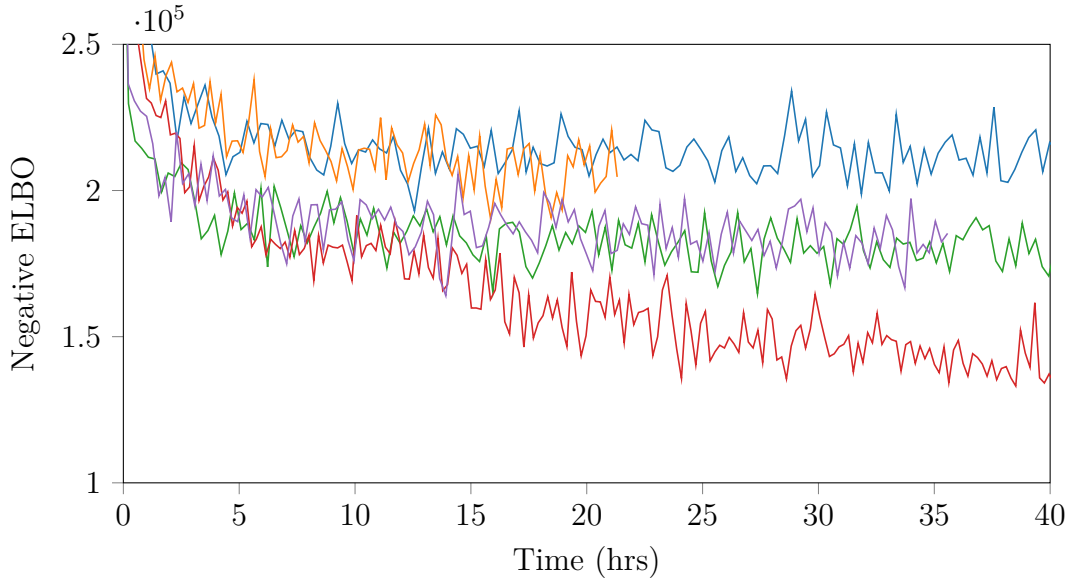


Fig. 5.15 Variational bound for CIFAR-10, using SE (blue), baseline weighted convolutional (orange), weighted colour-convolutional (green), additive colour-convolutional (purple) and multi-channel convolutional (red) kernels.

which allow for some form of decoupled response to different colours outperform the baseline weighted convolutional kernel. The final error rate of the multi-channel kernel the with multi-output inducing variable approximation was 38.1%, compared to 48.6% for the RBF kernel. While we acknowledge that this is far from state of the art using deep nets, it is a significant improvement over existing Gaussian process models, including the 44.95% error reported by Krauth et al. [2016], where a SE kernel was used together with their leave-one-out objective for the hyperparameters[4].

5.9 Comparison to convolutional neural networks

Just as Gaussian processes are infinite limits of single-layer neural network [Neal, 1994], convolutional Gaussian processes are a particular limit of single-layer convolutional neural networks. Single layer convolutional neural networks convolve filters $\mathbf{w}_h$ over an image, followed by a non-linearity $\phi(\cdot)$ to form the hidden representation. Each filter is shaped as a patch, and there are H filters, giving $\mathbf{w}_h \in \mathbb{R}^{w \times h \times H}$. After the hidden layer, all features are multiplied and summed with the weights $\mathbf{w}_f \in \mathbb{R}^{P \times H}$, to form a single scalar. We illustrate this in figure 5.16.

[4]It may be possible that the gain in performance seen in Krauth et al. [2016] from using the leave-one-out objective will also carry through to convolutional kernels.

The core assumption that leads to the convolutional Gaussian process, is to constrain $\mathbf{w}_f$ to use the same weights to sum spatially, and the same weights to sum over the H filters. This gives:

$$[\mathbf{w}_f]_{ph} = w_p w_g \,. \tag{5.65}$$

Given this assumption, we can perform the spatial summing after the summing over features. We identify result of the summation over w_g as outputs of the patch response function. If we place a Gaussian prior on w_g, and take the limit of $H \to \infty$, we obtain exactly the same situation as discussed by Neal [1994], where the patch response function becomes a Gaussian process. As in the convolutional Gaussian process, we do not place a prior over w_p, and let them remain as hyperparameters.

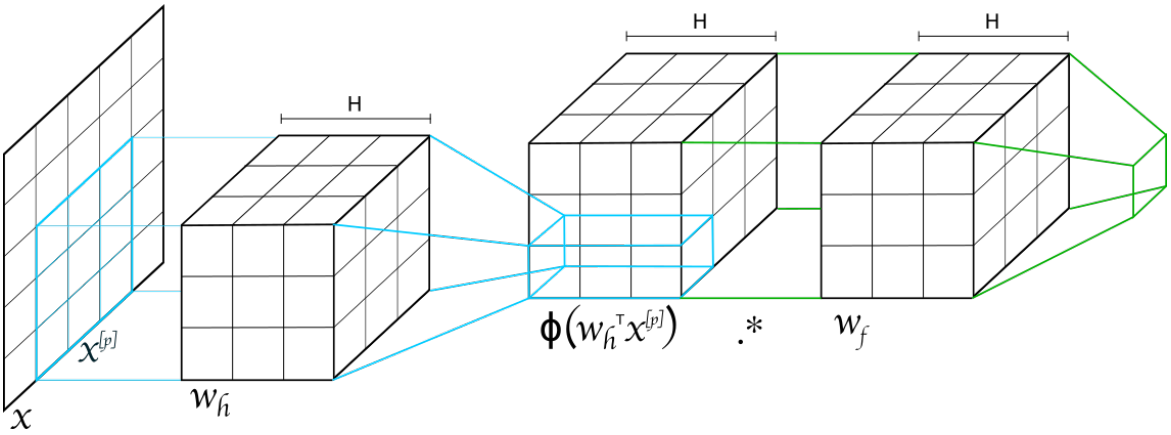


Fig. 5.16 Pictorial representation of the convolution required in stationary kernels. A single dot product with a patch is highlighted.

This comparison is useful to highlight one main difference between convolutional GPs and CNNs. Adding filters in a CNN *does* increase the capacity of the model, while it does not for the convolutional GP, as it already has an infinite number. If we were to do this, we would simply recover an additive kernel for the patch-response function.

5.10 Notes on implementation

A large bottleneck for the implementation is summation of kernel evaluations over numerous patches. A general implementation could simply extract the patches from the image, compute the kernel, and sum:

$$[K_{\mathbf{fu}}]_{nm} = \sum_p k_g(\mathbf{x}_n^{[p]}, \mathbf{z}_m) \,. \tag{5.66}$$

This can be implemented as evaluating a large $PN \times M$ kernel matrix, reshaping to $P \times N \times M$, and summing over the first dimension. For general kernels, this is required. However, if the kernel is stationary, i.e. $k_g(\mathbf{p}, \mathbf{p}') = k_g(|\mathbf{p} - \mathbf{p}'|)$, the first step is computing the matrix of pairwise distances between all patches and inducing points. If the patches were general inputs, this would be unavoidable. However, in this case neighbouring inputs overlap strongly, since they're all patches from the same image. By expanding the Euclidean distance, we can find the convolution operation (figure 5.17):

$$(\mathbf{x}_n^{[p]} - \mathbf{z}_m)^2 = \mathbf{x}_n^{[p]\mathsf{T}}\mathbf{x}_n^{[p]} - 2\mathbf{x}_n^{[p]\mathsf{T}}\mathbf{z}_m + \mathbf{z}_m^{\mathsf{T}}\mathbf{z}_m \,. \tag{5.67}$$

The inner product of $\mathbf{z}_m$ along all patches of $\mathbf{x}$ is a convolution operation. Additionally, the inner product of the patches with themselves is also a convolution of the squared image with a window of ones. This allows the Euclidean distance to be computed in $\mathcal{O}(\log E)$ rather than $\mathcal{O}(E)$. Additionally, much effort has gone into optimising implementations of convolutions (e.g. TensorFlow [Abadi et al., 2015] provides `tf.conv2d()`), together with placing them on fast hardware like GPUs thanks to the popularity of convnets.

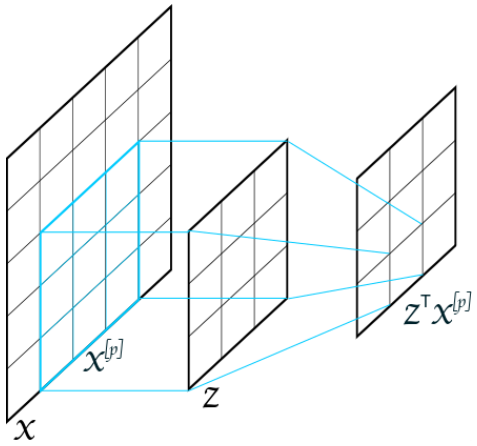


Fig. 5.17 Pictorial representation of convolution required in stationary kernels. A single dot product with a patch is highlighted.

5.11 Conclusions

We started this chapter by noting that in some cases, the performance of Gaussian processes may be held back as much by the lack of kernels with interesting generalisation properties, as the availability of accurate approximate inference schemes. We showed

that incorporating invariances into Gaussian process priors can be very helpful for improving generalisation, and is favoured by the marginal likelihood if it does.

Based on this understanding, we specifically looked at integrating convolutional structures (similar to those known in neural networks) into a Gaussian process model, in order to incorporate invariances beneficial for modelling functions on high-dimensional image inputs. For efficient inference, we introduced a matched inter-domain approximation, where inducing points were placed in the space of patches, leading to a procedure with computational similarities to convolutional neural networks. Several variations of convolutional kernels were applied to the standard MNIST and CIFAR-10 benchmarks, both of which are considered challenging for GPs, and strongly improved compared to previous GP models.

One particularly interesting observation was that using convolutional structure alone did not result in optimal performance. Although the convolutional kernel generalises well due to its constrained nature, its constrained nature also left certain systematic residuals unmodelled. By adding a universal squared exponential kernel to the model, and using the marginal likelihood approximation to weigh the magnitudes of both components, we could improve performance even further. This property, the ability to search over a space of different models and trade off data fit and complexity, is arguably the main promise for Gaussian process models for the future. Currently, deep learning models require trial-and-error and cross-validation to find the architecture. This process could be greatly simplified if hyperparameter selection could be guided by a differentiable objective function, as the marginal likelihood is for Gaussian process models.

It has to be emphasised though, that despite their elegance, Gaussian processes do lag behind deep neural networks in terms of performance. In this particular work, depth is the obvious missing ingredient. However, progress in deep Gaussian processes may open the door to powerful models which combine the benefits of both worlds, with convolutional Gaussian processes providing a starting point for layers useful for image processing tasks.

Chapter 6

New insights into random-input Gaussian process models

In this chapter, we want to deal with some conceptual issues surrounding Gaussian process models with random variables as inputs, with a particular focus on the the Gaussian Process State Space Model (GPSSM). The insights in this chapter were developed together with Richard Turner, Carl Rasmussen, Thang Bui and Roger Frigola[1]. My main contribution was the connection between the marginalised presentation presented by Frigola [2015], and the derivation using the process view. We argue that the presentation of GPSSMs (and to a lesser extent all random variable input GP models) is complicated by a conflation of random variables *on* the GP, and ones that are the *evaluation* at some random input location. We can also greatly simplify the derivations by viewing the models (informally) from the stochastic process perspective [Matthews et al., 2016], and resolve ambiguities in the graphical models that have been used to demonstrate GPSSMs in recent years. Over the next few sections we will build up to the GPSSM from GP regression, highlighting where the process view comes in at each stage. This new presentation has three main advantages over previous ones. First, the notation is clear, with unambiguous conditional distributions and graphical models. Second, the derivation of variational inference is greatly simplified, and third, the mean-field assumption in the approximate posterior between the transition function and the states becomes clear. The main drawback of this presentation is some slight abuse of notation regarding the infinite-dimensional Gaussian process.

[1]The insights were written up and circulated as a note around the group. Some of the insights were discussed in passing in Frigola [2015], where the note was cited.

6.1 Issues with augmentation

We first want to briefly justify the need for the "approximating process" view in random-input GPs by considering some issues that can arise from the augmentation view.

In section 2.4.2 we discussed how the the KL divergence between latent processes can be minimised to find a sparse approximation. Titsias [2009b] originally presented the variational inference scheme starting from the formulation with $f(\cdot)$ marginalised out. The model is then "augmented" with the inducing variables $\mathbf{u}$, by specifying a joint distribution $p(\mathbf{f}, \mathbf{u})$ which is marginally consistent with $\mathbf{f}$. Variational inference is then performed over both $\mathbf{f}$ and $\mathbf{u}$. While in Titsias's [2009b] work, the augmenting was done in a way that led to minimising the KL between processes, not every augmentation guarantees a good approximation, and the augmentation argument by itself does not provide an indication of what a good augmentation is[2].

The main worry about the augmentation argument alone (also voiced by Richard Turner), is that as soon as the model is augmented with arbitrary variables, the variational inference scheme is obliged to approximate the posterior over both [Matthews, 2016, see §3.4.3]. We can easily construct augmentations which lead to poor approximations. For example, if we chose $\mathbf{u}$ to be noisy observations of the latent process at locations Z. We can explicitly represent the noise variables (figure 6.1) to emphasise the change in model, even when the model is marginally consistent. As before, we end up minimising the KL divergence

$$\mathrm{KL}[\, p(\mathbf{f}|\mathbf{u})q(\mathbf{u}) \,\|\, p(\mathbf{f}, \mathbf{u}|\mathbf{y}) \,]\,, \tag{6.1}$$

only this time, with the posterior

$$p(\mathbf{f}, \mathbf{u}|\mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} K_{\mathbf{fy}} \\ K_{\mathbf{uy}} \end{bmatrix} K_{\mathbf{yy}}^{-1}\mathbf{y}, \begin{bmatrix} K_{\mathbf{ff}} & K_{\mathbf{fu}} \\ K_{\mathbf{uf}} & K_{\mathbf{uu}} + \sigma_\epsilon I \end{bmatrix} - \begin{bmatrix} K_{\mathbf{fy}} \\ K_{\mathbf{uy}} \end{bmatrix} K_{\mathbf{yy}}^{-1} \begin{bmatrix} K_{\mathbf{fy}}^\intercal & K_{\mathbf{uy}}^\intercal \end{bmatrix} \right). \tag{6.2}$$

The bound we get from this KL can now never be tight, since the diagonal term in the covariance of $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$ can never be represented with our approximate posterior. The approximate posterior now needs to trade off accurately representing this noise term (which we do not care about) and accurately capturing the distribution over $\mathbf{f}$ (which we do care about). We therefore want to revisit derivations of uncertain input GPs with the "approximating process" view, to verify their correctness as well.

[2] Although Hensman and Lawrence [2014] discusses justifications for good augmentations in terms of information theory.

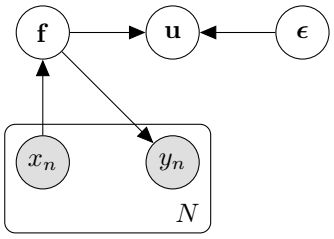


Fig. 6.1 A model that is marginally consistent, but that does *not* result in a variational approximation with the KL between processes as the minimised quantity.

6.2 Conditioning in GP regression

We first consider the standard GP regression scenario, where we change notation slightly. We are given some inputs $x_{1:N} = \{x_n\}_{n=1}^N$ in the space $\mathcal{X}$ and corresponding outputs $y_{1:N} = \{y_n\}_{n=1}^N$, and want to make predictions over future inputs x^*. We place a Gaussian process prior over the latent function. The usual Gaussian likelihood depends on the latent function at the input given in the training set:

$$f \sim \mathcal{GP}(f; 0, K) \tag{6.3}$$

$$y_n | f(\cdot), x_n \sim \mathcal{N}\left(y_n; f(x_n), \sigma_y^2\right). \tag{6.4}$$

The Gaussian process is a measure on the space of functions $\mathbb{R}^{\mathcal{X}}$, which we can't strictly represent as a density. However, it behaves like an "infinitely long vector" in the sense that if we want the marginal over any finite set of inputs $\tilde{X} \subset \mathcal{X}$, we obtain Gaussian distributions (section 1.2.4). Because the likelihood only depends on a finite set of points, we can focus on the posterior of these only, and marginalise out all other random variables on the Gaussian process that do not correspond to an input point[3]. We can calculate the posterior:

$$p(f(x_{1:N})|y_{1:N}) = \frac{p(y_n|f(\cdot), x_n)p(f(x_{1:N}))}{p(y_{1:N})}. \tag{6.5}$$

We use $f(x_{1:N})$ to denote the marginal distribution of the GP at the locations $x_{1:N}$. Only the likelihood is conditioned on the training inputs, as this determines which function values it depends on. The prior is *not* conditioned on the training inputs, we simply use the observed values to determine which marginals we choose to represent in our calculations. This is consistent with how we defined the model. After all, the

[3]As discussed in section 1.2.3, we only need this posterior to recover the full latent Gaussian process posterior.

prior distribution over the latent *function* does not change with knowledge of only the training inputs. For a more specific example, consider the distribution of the function value at the input 3, assuming $\mathcal{X} = \mathbb{R}$ for now. $f(3)$ is a random variable with a distribution that remains unchanged after gaining the knowledge that our y_ns will be observed at X.

This presentation is subtly different to how the model has been presented in the past. Rasmussen and Williams [2005], for example, condition the prior over function values on the inputs, writing $p(\mathbf{f}|X)$. We can arrive at this formulation by integrating out the Gaussian process:

$$p(f) = \mathcal{GP}(f; 0, K) \tag{6.6}$$

$$p([\mathbf{f}]_n|f, x_n) = \delta(f(x_n) - [\mathbf{f}]_n) \tag{6.7}$$

$$p(y_n|[\mathbf{f}]_n) = \mathcal{N}\left(y_n; [\mathbf{f}]_n, \sigma_y^2\right). \tag{6.8}$$

In this case, we construct the random variable $\mathbf{f}$ to be the collection of the function $f(\cdot)$ evaluated at the training input locations (we denote the nth element of $\mathbf{f}$ using $[\mathbf{f}]_n$ to explicitly avoid confusion with f). We also change the likelihood to not depend directly on the latent function $f(\cdot)$, but on $\mathbf{f}$ instead. We can again integrate out all unnecessary random variables, which in this case is the entire Gaussian process, resulting in a distribution of $\mathbf{f}$ which *is* conditioned on X.

$$p(\mathbf{f}|X) = \mathcal{N}(\mathbf{f}; 0, K_{\mathbf{ff}}) \tag{6.9}$$

Conditioned on X, the latent Gaussian process can be recovered by finding the joint distribution of $\mathbf{f}$ with an arbitrary marginal $f\left(x^*_{1:\tilde{N}}\right)$, and integrating the rest of f out.

Although both these models are equivalent interpretations, the separation between variables *on* the GP and *evaluations* will become important when we consider random inputs. We can view the graphical models for each option in figure 6.2.

6.3 Adding uncertainty on the inputs

The GPLVM is the canonical example of a GP model with uncertainty on its inputs, and Titsias and Lawrence [2010] developed the first variational inference scheme over both the GP mapping and the inputs X (see Damianou et al. [2016] for a comprehensive and modern review of the Bayesian GPLVM and derivative models). The distinction

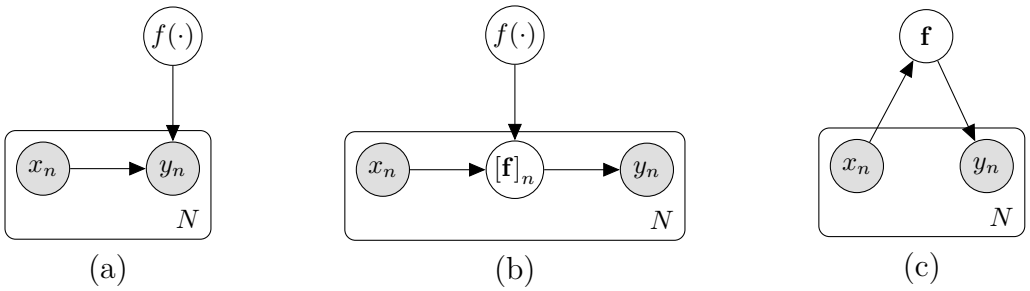


Fig. 6.2 Three equivalent graphical models for GP regression, or the GPLVM if we consider the x_n variables to be unobserved. Left: The likelihood depends directly on the latent process $f(\cdot)$. Middle: Explicit representation of the evaluation variable $\mathbf{f}$ is added. Right: The GP is integrated out, and the evaluations $\mathbf{f}$ become correlated.

between $\mathbf{f}$ and $f(x_{1:N})$ becomes more apparent in this case. The graphical model for the GPLVM is the same as in figure 6.2, only with $\{x_n\}$ being unobserved.

It is now problematic to refer to a random variable on the Gaussian process as $f(x_n)$, particularly if we refer to a joint as $p(x_n, f(x_n))$. The value of x_n determines the *identity* of the random variable $f(x_n)$ (i.e. which variable on the GP we're considering). This means that the notation $p(x_n, f(x_n))$ is nonsensical, as it does not refer to the density of a well-defined random variables. This issue is an infinite-dimensional analogue to issues with representing mixture models as graphical models. Mixture models share the feature that the mixture indicator determines which cluster mean determines the mean of a new data point. In regular directed graphical models, there is no choice but to make the data depend on the mixture indicator as well as all cluster means[4].

It should be clear now that the likelihood $p(y_n|f(\cdot), x_n)$ really does need to depend on the *entire* latent Gaussian process: For a stochastic input, it is not clear which random variable on the GP is required until x_n is observed – it could depend on any. Explicitly representing the evaluation variable $\mathbf{f}$ as the evaluation of the GP at a random input location solves this problem. We can correctly refer to the joint density $p(x_n, [\mathbf{f}]_n)$, and just like in regression, it is clear that the conditional $\mathbf{f}|X$ is Gaussian.

While the specification of the finite dimensional model is correct, it obscures the connection to the latent process when performing inference. We gain two insights from considering the process view: how to alternately sample the GP and inputs, and that the variational approximation from Titsias and Lawrence [2010] was in fact mean-field between the GP and the inputs.

[4]Minka and Winn [2009] introduce "gates" as an extension to directed graphical models to visually describe this situation more elegantly.

6.3.1 Gibbs sampling

Consider the task of performing Gibbs (or, more accurately, Metropolis-within-Gibbs) sampling in the marginalised formulation of the GPLVM. Given that we have the joint $p(X, \mathbf{f}, \mathbf{y})$, it would be natural to alternately sample from $p(\mathbf{f}|X, \mathbf{y})$ and $p(X|\mathbf{f}, \mathbf{y})$. The former conditional of $\mathbf{f}$ can be sampled from exactly, while $p(X|\mathbf{f}, \mathbf{y})$ has to be handled with Metropolis-Hästings, giving the acceptance probability:

$$a = \min\left(1, \frac{p(X_{prop}, \mathbf{f}, \mathbf{y})}{p(X, \mathbf{f}, \mathbf{y})} \frac{q(X_{prop} \to X)}{q(X \to X_{prop})}\right) \tag{6.10}$$

Following this procedure, we may expect the latent GP to remain in the same state during the update of X. This is not true, since $\mathbf{f}$ are not variables *on* the GP, but observations of the GP at X. If X is re-sampled, we are in effect also constraining the latent GP to have the old $\mathbf{f}$ at the new locations, leading to a different predictive distribution at other points.

If we consider the latent process, we arrive at a different procedure for $p(X|\mathbf{f}, \mathbf{y})$. Imagine if we had truly sampled the full $f(\cdot)$. Keeping this variable constant would entail looking up a new value for $\mathbf{f}$ after proposing the new inputs. Sampling the infinite-dimensional $f(\cdot)$ is impractical, of course. However, *retrospective sampling* offers an elegant solution. Papaspiliopoulos and Roberts [2008] proposed to switch the order of sampling of the infinite-dimensional object with values that depend on it. Adams et al. [2009] used this retrospective sampling trick in a Gaussian process model, where it simply amounts to sampling the GP inputs before their sampling respective function values from the GP conditioned on all previously sampled points[5]. In the limit where we instantiate an infinite number of function values, this process would become a simple look up. Following this idea, our procedure for re-sampling X now becomes:

1. Sample $X_{prop} \sim q(X \to X_{prop})$,
2. Sample $f(X_{prop}) \sim p(f(X_{prop}|f(X), \mathbf{y})$,
3. Accept with

$$a = \min\left(\frac{p(\mathbf{y}|f(X_{prop}))\cancel{p(f(X_{prop})|f(X))}}{p(\mathbf{y}|f(X))\cancel{p(f(X)|f(X_{prop}))}} \frac{\cancel{p(f(X)|f(X_{prop}))}q(X_{prop} \to X)}{\cancel{p(f(X_{prop})|f(X))}q(X \to X_{prop})}\right). \tag{6.11}$$

[5]Intuitively this is akin to lazy evaluation – variables are only sampled once they are needed.

Crucially, marginalised over this transition, the distribution of $f(\cdot)$ at any collection of fixed points Z remains unchanged, which is what we expect from a Gibbs sampler. The next sampling step would re-sample $f(\cdot)$ by simply re-sampling $f(X)$. A full description of the retrospective sampling trick for GPs can be found in Adams et al. [2009]. The main importance here is that by considering the marginalised model, we (perhaps inadvertently) mix up inference over the uncertain inputs X, and the underlying GP $f(\cdot)$. The process view avoids this. A similar issue occurs in variational inference.

6.3.2 Variational inference

The original derivations of the GPLVM in Titsias and Lawrence [2010] start with the evaluation variables $\mathbf{f}$ after the GP has been integrated out. The derivation of the variational inference scheme then required putting variables on the GP back in, which was presented as the "augmentation trick". The variational distribution that was proposed was:

$$q(\mathbf{f}, \mathbf{u}, X) = q(X)q(\mathbf{u})p(\mathbf{f}|\mathbf{u}, X)\,. \tag{6.12}$$

At first sight, this method does factorise X and $\mathbf{u}$, but does allow dependence between X and $\mathbf{f}$. This makes it seem as though there are dependencies between X and the Gaussian process. Viewing the approximation from the process view, we see that the approximation does, in fact, factorise X and $f(\cdot)$.

In the following alternative derivation, we focus directly on the marginals of the GP, instead of $\mathbf{f}$. With some abuse of notation, we write the GP prior as $p(f)$, even though there is no density w.r.t. the Lebesgue measure. We do this, on the understanding that we will integrate out all variables except a finite set. This allows us to write down the marginal likelihood:

$$\begin{aligned}\log p(y_{1:N}) &= \log \int \prod_{n=1}^{N} p(y_n|f, x_n)p(f)p(X)\mathrm{d}f\mathrm{d}X \\ &= \log \iint \prod_{n=1}^{N} p(y_n|f(x_n))p(f(X))\mathrm{d}f(X)p(X)\mathrm{d}X\,. \end{aligned} \tag{6.13}$$

As mentioned earlier, we can not write a joint distribution between the inputs and the function values on the GP like $p(X, f(X))$, as the identity of the random variable $f(X)$ would be implied to change for different values of X. However, the inner integral, and all the densities in it, are well defined, since the value for X is fixed in the outer integral. We write the approximate posterior factored as (again with some abuse of

notation):

$$q(f, X) = q(X)q(\mathbf{u})p(f_{\neq \mathbf{u}}|\mathbf{u})\,. \tag{6.14}$$

With this, we mean that we can get the marginal at any set of input locations $\tilde{X}$ (distinct from the X used in $q(X)$) using the distribution of a GP conditioned on $\mathbf{u} = f(Z)$:

$$q(f(\tilde{X}), X) = q(X) \int q(\mathbf{u})p(f(\tilde{X})|\mathbf{u})\mathrm{d}\mathbf{u}\,. \tag{6.15}$$

We can now create a variational lower bound in the same way as in section 2.4.2, by explicitly representing the same variables in the prior and posterior, and noting that the conditional for any extra variables cancel.

$$\begin{aligned}
\log p(y_{1:N}) &= \log \int q(X) \Bigg[\int q(f(X), f(X^*), \mathbf{u}) \\
&\qquad \prod_{n=1}^{N} \frac{p(y_n|f(x_n))\cancel{p(f(X), f(X^*)|\mathbf{u})}p(\mathbf{u})}{q(X)\cancel{p(f(X), f(X^*)|\mathbf{u})}q(\mathbf{u})} \mathrm{d}f(X)\mathrm{d}f(X^*)\mathrm{d}\mathbf{u} \Bigg] \mathrm{d}X \\
&\geq \int q(X) \left[\int p(f(X)|\mathbf{u})q(\mathbf{u}) \log \frac{p(y_n|f(x_n))p(\mathbf{u})}{q(X)q(\mathbf{u})} \mathrm{d}f(X)\mathrm{d}\mathbf{u} \right] \mathrm{d}X \\
&= \mathbb{E}_{q(x_n)} \Big[\mathbb{E}_{q(f(x_n)|\mathbf{u})q(\mathbf{u})} [\log p(y_n|f(x_n))] \Big] - \mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u}) \,] \\
&\qquad - \mathrm{KL}[\, q(X) \,\|\, p(X) \,]
\end{aligned} \tag{6.16}$$

This alternative derivation shows the mean-field property of the variational distribution, and highlights that all the variables involved are actually on the GP.

6.4 Gaussian Process State Space Models

GPSSMs have recently been studied in various forms with various inference techniques. Progress in the development of inference for the model, however, has outpaced the clarity of presentation, with papers containing mathematically ambiguous notation and graphical models with unclear semantics, despite the methods being algorithmically correct. Again, we argue that the root cause is that there is not enough clarity about the process that the finite dimensional marginals come from. Much of the clarity here was first presented by Frigola et al. [2014]. Here we aim to continue the work, and revisit the GPSSM in the same way as above with the GPLVM. This simplifies

the derivation of the lower bound, highlights mean-field properties of the variational distribution and clears up the graphical model.

6.4.1 Model overview

The GPSSM posits that observed time series $y_{1:T}$ can be explained as noisy observations[6] from some latent time series $x_{1:T}$ with a Markov structure. The transition from one latent state to the next is determined by some non-linear function $f(\cdot)$ given a GP prior:

$$f \sim \mathcal{GP}(0, k(\cdot, \cdot)) \,, \tag{6.17}$$

$$x_0 \sim \mathcal{N}(x_0; 0, \Sigma_0) \,, \tag{6.18}$$

$$x_{t+1}|f, x_t \sim \mathcal{N}(x_{t+1}; f(x_t), \Sigma_x) \,, \tag{6.19}$$

$$y_t|x_t \sim \mathcal{N}(y_t; x_t, \Sigma_y) \,. \tag{6.20}$$

Although the model is concisely described by referencing the underlying transition function $f(\cdot)$ which makes the whole system Markov, presenting the model using finite dimensional marginals is complicated. The model requires feeding the output of the GP back into itself as an input. This turns the mild issues that existed in the GPLVM concerning the identity of random variables, and the question of what variables are actually conditioned on others, into real conceptual issues. Early papers like Wang et al. [2006] and Ko and Fox [2009] note that the problem can be seen as a regression problem from $x_{0:T-1}$ to $x_{1:T}$. Directly substituting this into the prior for GP regression as in equation 6.9 would lead to the nonsensical expression $p(x_{1:T}|x_{0:T-1})$ where we both want to describe a distribution over a subset of the same variables that we condition on. This was noted in Wang et al. [2006] as an unexplained oddity.

This conceptual problem stems from considering the random variables of the latent state as being the same as the random variables on the GP. In the GPSSM, which variable on the GP is required for describing the distribution of x_{t+1} depends on the value that x_t takes – the previous output of the same GP. Frigola et al. [2014] presents a view that solves many of these issues by explicitly splitting the evaluations of the GP and states into different variables. The full model ("probability of everything") was then constructed incrementally, sampling from the GP one value at a time.

[6] Other observation models can also be considered, see Frigola [2015] for a great overview.

6.4.2 Graphical model

Here we will investigate an unambiguous graphical model for the GPSSM, and how it relates to the graphical model when only finite dimensional marginals are represented. In the past, GP models have relied on the "thick black bar" notation, used in Rasmussen and Williams [2005] to indicate variables on the same GP (and hence from a fully pair-wise connected factor graph). Frigola et al. [2013] and Frigola [2015] present the GPSSM in this way as well (figure 6.3). There are several issues with this illustration. Firstly, the arrow from an x_n to an f_{n+1} does not indicate a conditioning. As discussed earlier, the GP prior does not change with knowledge of the input locations. What is meant, is that x_n picks out the appropriate variable from the GP. Secondly, the f_ns are not marginally Gaussian, which one would expect from variables on a Gaussian process. Finally, the graphical model does not obey the usual rules of conditioning in directed graphical models [Shachter, 1998]. For example, observing f_2 and f_1 does not make f_3 independent of x_1 and x_0.

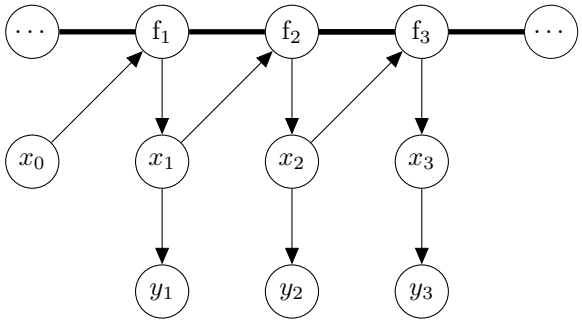


Fig. 6.3 GPSSM using the "thick black bar" notation.

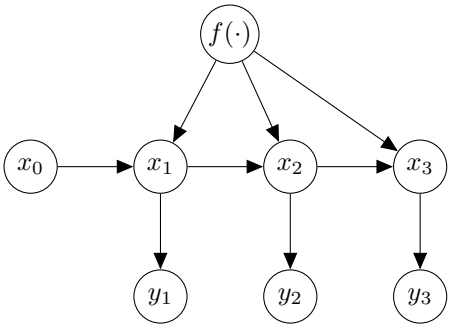


Fig. 6.4 GPSSM with explicit reference to the full latent function.

The graphical model including the full non-parametric model (figure 6.4) does fulfil these requirements. We can, however, recover the graphical model for the finite dimensional presentation of the model [Frigola et al., 2014] by introducing the evaluation variables (f_t – not the curly f) again, and then integrating out $f(\cdot)$. The generative model becomes:

$$\begin{aligned} f &\sim \mathcal{GP}(0, k(\cdot,\cdot)) & x_0 &\sim \mathcal{N}(0, \Sigma_0) \\ \mathrm{f}_t|x_{t-1} &\sim \delta(\mathrm{f}_t - f(x_{t-1})) & x_{t+1}|\mathrm{f}_t &\sim \mathcal{N}(\mathrm{f}_t, \Sigma_x) \\ y_t|x_t &\sim \mathcal{N}(x_t, \Sigma_y) \end{aligned} \tag{6.21}$$

By carefully integrating out f, we return back to the formulation from Frigola et al. [2014] (appendix C).

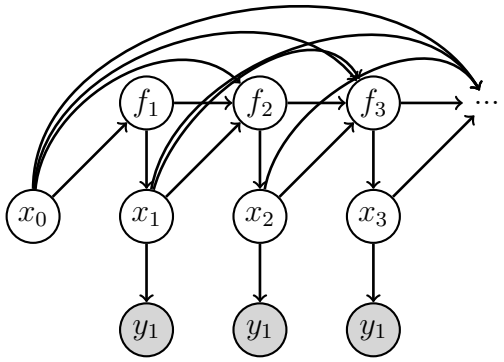


Fig. 6.5 Resulting graphical model after marginalising out the infinite dimensional f in figure 6.4.

6.4.3 Variational inference

Finally, we show how the process view can greatly simplify the derivation of the variational inference lower bound, while avoiding the augmentation trick. Similar to the derivation for the GPLVM in equation 6.16, we initially instantiate the full latent process, and show that our bound only ever relies on well-defined expectations. The derivation here was proposed by Richard Turner. We start again informally with an integration over f. We again also do the integration over $x_{0:T}$ last, so we can refer to densities of $f(x)$, when they are well-defined:

$$\begin{aligned}\log p(y_{1:T}) &= \log \int p(x_0) \prod_{t=1}^{T} p(y_t|x_t) p(x_t|x_{t-1}, f) p(f) \mathrm{d}f \mathrm{d}x_{0:T} \\ &= \log \int p(x_0) \prod_{t=1}^{T} p(y_t|x_t) p(x_t|f(x_{t-1})) p(f(x_{0:T}), \mathbf{u}) \mathrm{d}f(x_{0:T}) \mathrm{d}\mathbf{u} \mathrm{d}x_{0:T} \end{aligned} \quad (6.22)$$

Our variational distribution is the same as for the GPLVM in equation 6.14. Again, we can represent any number of other variables on the GP without changing the bound:

$$\log p(y_{1:T}) \geq \int p(X) \Bigg[\int q(f(X), f(X^*), \mathbf{u}) \log \frac{p(x_0) \prod_{t=1}^{T} p(y_t|x_t) p(x_t|f(x_{t-1})) \cancel{p(f(X), f(X^*)|\mathbf{u})} p(\mathbf{u})}{\cancel{p(f(X), f(X^*)|\mathbf{u})} q(\mathbf{u})} \mathrm{d}f(X) \mathrm{d}f(X^*) \mathrm{d}\mathbf{u} \Bigg] \mathrm{d}x_{0:T} \tag{6.23}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{q(x_t)}[\log p(y_t|x_t)] + \sum_{t=1}^{T} \mathbb{E}_{q(x_{t-1:t})} \Big[\mathbb{E}_{p(f(x_{t-1}), f(x_t)|\mathbf{u}) q(\mathbf{u})}[\log p(x_t|x_{t-1}, f)] \Big] + \mathcal{H}(q(x_{0:T})) - \mathrm{KL}[\, q(\mathbf{u}) \,\|\, p(\mathbf{u}) \,] + \mathbb{E}_{q(x_0)}[\log p(x_0)] \tag{6.24}$$

The expectations required for computing this bound are well described by Frigola [2015], when sampling from the optimal $q(x_{0:T})$ and McHutchon [2014], when optimising a Gaussian $q(x_{0:T})$.

6.5 Conclusion

We presented alternate derivations of the GPLVM and GPSSM, which more closely highlight the connection to the "approximating process" view of variational inference for GPs. It shows that the models can be derived without the use of the augmentation argument, relying only on not marginalising out certain variables. This view has pitfalls when considering random inputs. It is not possible to write down a well-defined joint distribution between variables on the GP and the random inputs. However, when performing the integrals carefully in the correct order, we can work with well-defined densities. While the difference to the previous derivations is subtle, the derivations presented here are useful in two ways. They are simpler to derive, as they don't require the incremental construction of the GP, and they highlight the independence assumptions in the approximate posterior more clearly.

Chapter 7

Discussion

So far in this thesis, we have discussed our contributions within the context of the Bayesian framework and the study of Gaussian process models. In this final chapter, we will take a broader look at the field, discuss the wider implications of this work, and take the liberty to speculate about possible future implications. We will particularly address the question of what the advances in Gaussian processes made in this thesis can offer large-scale applications currently dominated by deep learning.

7.1 Non-parametrics and modern deep learning

Machine learning has seen enormous growth in recent years in terms of interest from both private and public institutions and the general public, largely due to impressive improvements in performance on academic benchmarks and real-world industrial problems. Deep learning has been a large contributor to this growth, due to its ability to leverage large datasets to learn complex non-linear relationships for a wide variety of tasks like image recognition and natural language translation [LeCun et al., 2015]. In the resulting wave of enthusiasm about deep learning, it seems that the performance gap of Bayesian non-parametric models has become well-known, causing their advantages[1] to be under-emphasised as well.

There have been various attempts to bring the benefits of the Bayesian methodology to neural networks over the years (e.g. MacKay [1992b,a]; Neal [1994]; Kingma et al. [2015]; Gal and Ghahramani [2016], see Gal [2016, §2.2] for a good review). Modern approaches are required to scale to the large datasets that are common in deep learning, and mainly focus on obtaining estimates of posterior uncertainty. This approach has

[1] Uncertainty estimates, objective functions for hyperparameter selection, and coherent models for increasing dataset sizes (section 1.3).

resulted in improved (and already useful) uncertainty estimates compared to maximum likelihood methods.

However, current Bayesian deep learning approaches mainly focus on approximating the posterior on the weights only, with other benefits not yet being fully realised. We take a moment to consider the current gap between the promises of the Bayesian non-parametric framework (section 1.3) and current deep learning practice.

- **Robust uncertainty.** While approximate weight posteriors have improved deep learning uncertainty estimates, it is still an open question as to how well the true posterior is actually captured. More fundamentally though, current methods still only approximate *parametric* deep learning models. In section 1.2.5 we argued that having "enough" basis functions in the model (with infinity being a convenient way to have enough) was important for obtaining robust uncertainties, particularly away from the input data distribution. Perhaps non-parametric layers can offer further improvements. This may be of particular importance when dealing with adversarial examples [Szegedy et al., 2013], as uncertainty has been shown to provide additional robustness [Li and Gal, 2017].

- **Hyperparameter selection.** Deep learning models generally use cross-validation for tuning hyperparameters. The marginal likelihood (or approximations to it) provides an elegant alternative. The ability to maximise the marginal likelihood using gradients (in the same framework as backpropagation) may provide a faster and more convenient way of training models. Currently, no approximations to the marginal likelihood of deep neural networks that can be used for hyperparameter selection exist.

- **Continual learning.** Deep learning models struggle in settings where increasing amounts of data become available. Small models are preferred for small datasets, with models becoming more complicated as more data gets available. This is at odds with the Bayesian non-parametric approach of using a single high-capacity model with appropriate complexity controls.

Deep Gaussian processes [Damianou and Lawrence, 2013] may provide a framework for combining the best of both worlds, by using non-parametric Gaussian processes as layers in deep models. Deep GPs still require significant development before they can considered to improve on current deep learning, with the two usual questions in Bayesian modelling remaining open:

- How do we perform accurate inference?

- What needs to change in the model for it to be useful on datasets we care about?

Many different inference methods have been proposed, all of which are derived from a well-known single layer approximations, e.g. Random Fourier Features [Cutajar et al., 2017], EP [Bui et al., 2016], or variational inference [Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017]. However, all approaches start with the same model, using multiple independent GP priors for each layer.

It is within this context that this thesis is best understood, with us aiming to contribute new insight, inference methods, and models which can be used towards the goal of creating more flexible models with Gaussian processes.

7.2 Summary of contributions

In this thesis, we only consider inference and modelling using single-layer Gaussian process models. This is of importance, as we can not expect to create good deep models without good understanding, approximations, and modelling capabilities in single layer models. In this thesis, we broadly address four questions:

- What properties do we want from Gaussian process approximations? (Chapter 2)
- How well do current objective functions live up to our requirements? (Chapter 3)
- How can we better approximate Gaussian process models? (Chapters 4 and 5)
- How can we create better Gaussian process models? (Chapter 5)

Chapter 2 focuses on reviewing existing approximations in the light of some desiderata for approximations to non-parametric basis function models that we propose. While these desiderata have been implicit in much of the work on variational GP approximations we discussed, the goal of these approximations is not always clearly recognised in other works. We hope that stating these desiderata will also provide some guidance to how to assess new approximations. Following this with chapter 3, we use our desiderata to perform an in-depth investigation of the two popular FITC and VFE objective functions for selecting approximate posteriors. We show that FITC is biased away from solutions that accurately approximate the model, despite its (often) good performance in testing metrics. This serves as an example that assessing whether approximate Bayesian inference is truly accurate, we need more empirical assessment than test set performance.

Chapter 4 and 5 both investigate using inter-domain inducing variables for representing posteriors. In chapter 4, we aspired to the common goal of improving the approximation of a kernel with very general properties (squared exponential), in our case with a more flexible posterior that could add basis functions to the mean at a sub-cubic cost. Particularly in very resource constrained situations, our method yielded sparser, more compressed approximations. In some cases, we noticed that in higher capacity regimes, the increased flexibility in the approximate posterior was not utilised fully, and any improvement was marginal over standard methods due to poor hyperparameter fitting.

More use can be gained from inter-domain inducing variables if they are tailored to the model they are approximating. In chapter 5 we used specially designed inter-domain inducing variables to cheaply and accurately approximate kernels with invariances, notably convolutions. Convolutions have been widely utilised in neural networks for improving generalisation performance on image data, but until now, have not been used within the GP framework. We showed an improvement from 2.0% to 1.17% classification error on MNIST compared to earlier work using the squared exponential kernel. The matched inter-domain approximation was crucial for obtaining a computationally practical method. One result that is particularly appealing in light of the aim of allowing differentiable hyperparameter selection (discussed in section 7.1) was the ability of the marginal likelihood to select weightings for convolutional and squared exponential kernels that improved performance.

Finally, in chapter 6, we discuss conceptual issues surrounding random input GP models. While correct, previous expositions of the GPLVM and GPSSM obscured the link to the underlying process. As a consequence, derivations were often complicated, and did not clearly highlight mean-field assumptions. Using the recent insights from the "process view" of variational inference in GP models, we simplify the derivations and clarified assumptions.

7.3 Future directions

We now briefly discuss some possible future directions for research based on the issues we address throughout this thesis. In section 7.1 we discussed the current pervasiveness of deep learning, and that a Bayesian non-parametric approach has potential to offer solutions to concrete problems with current deep learning systems. Consequentially, we will mostly discuss future work in this vein.

Objective functions for GP inference

While section 3.2 illustrated situations where EP/FITC does not behave as the model it is supposed to approximate, this does not imply that the alternative VFE approach is the only method worth investing future research efforts in. In our set-up, VFE was favourable for the same reason which makes it convenient in general: Since VFE is guaranteed to improve the approximation (in terms of KL) when given a more flexible class of approximate posteriors to choose from, a general recipe to improve it in failure cases is to widen the class of approximate posteriors. Optimising the inducing points therefore did not present any trouble. EP/FITC did not share this property, as the true posterior was not preferred when it was available.

However, we are generally interested in situations where the class of posteriors *is* constrained, with Gaussian process regression being perhaps slightly unusual due to sparse approximate GP posteriors being able to get very close to the true posterior. In non-sparse classification tasks, EP is known to perform very well [Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008]. Variational methods struggle due to the heavy penalty that the KL objective applies when the approximate posterior assigns mass where the true posterior does not.

Whether variational inference or EP will provide a future way forward seems to depend on the extent to which approximate posteriors that are flexible enough to make VFE work are computationally tractable. In situations where the required flexibility is too expensive, EP style methods may be able to provide a good limited approximation more quickly. This may be an increasingly important question for Bayesian deep models, where independence assumptions between layers are common.

Deep Gaussian processes

Given the impact that convolutional layers have had on deep learning practice, it is tempting to investigate whether convolutional GP layers can form a similar building block for deep GP models. The main modification that would be required, is the definition of an image to image Gaussian process, which can be achieved by not performing the pooling summation.

In chapter 5, we also used the marginal likelihood selecting a weighting between a convolutional and fully connected GP. While we only considered a single layer, the ability to do this automatically in deep models could significantly simplify the search for good architectures.

In section 7.1 we discuss uncertainty quantification and continual learning as additional benefits of the Bayesian framework. Demonstrating these capabilities requires significant extra work, particularly as these properties are most useful when integrated into larger systems.

Learning model invariances

We viewed the convolutional model discussed in chapter 5 as an interesting case within more general invariances. The inter-domain approach we introduced is equally applicable to kernels with generally invariant kernels. If we would be able to parameterise a large class of kernels with different invariances, we could use the marginal likelihood to select useful ones. This is a large opportunity, as invariances provide very strong and non-trivial generalisation biases. So far, the usefulness of the marginal likelihood has been limited as it is only used to select very simple properties of kernels (like lengthscale). Selecting invariances may provide a very strong use case for the Bayesian framework.

Computational properties of Gaussian processes

Despite the advances of sparse approximation methods, Gaussian process models are still at a strong computational disadvantage to neural networks. Even sparse methods still require the inversion of a matrix, which is often done using a Cholesky decomposition. Deep neural networks require only matrix multiplications, which are efficiently parallelisable, and require less numerical precision. The popularity of deep neural networks has also had the effect of modern computing hardware being tailored to their needs, with GPUs and TPUs increasingly focusing on high-parallelism, low-precision computations. This presents an extra challenge for future Gaussian process work, as current linear algebra based implementations require high-precision computation. Iterative matrix product based algorithms like Conjugate Gradients have been proposed in the past [Gibbs and MacKay, 1997] as a way to speed up GP inference.

Leveraging the advances in hardware made for deep learning is likely to be crucial for GPs to remain practically relevant. The convolutional GP already presents an example of how deep learning advances can be useful, due to the ability to leverage optimised GPU implementation of convolution operations section 5.10.

Optimisation

Similar to the development of hardware tailored to deep learning, significant effort in the deep learning community is expended on more efficient optimisation, with Adam [Kingma and Ba, 2014] being a prime example. Adam works very well for optimising neural networks, despite questions about convergence guarantees [Reddi et al., 2018]. The lack of convergence guarantees perhaps do not present as much of a problem for training neural network models, as for Gaussian processes. For models trained with maximum likelihood, not converging may be helpful to prevent overfitting. When training Gaussian process models we put a lot of effort into constructing objective functions which are less susceptible to overfitting, and so we ideally would want to train them to convergence. This raises the question of whether optimisation routines designed for neural networks may need modification to work well for Gaussian processes. One benefit of the variational inference, is that there are frameworks for understanding and improving optimisation behaviour. Natural gradients have been used with success in the past [Hensman et al., 2013], and perhaps may provide a useful direction in the future.

Software

Finally, a large contributor to the uptake of methods in the community, is the availability of easily usable software packages. The growth in both deep learning research and practice has been largely influenced by very popular software packages, such as Theano [Theano Development Team, 2016], Caffe [Jia et al., 2014], and Keras [Chollet et al., 2015]. Providing good software is a research challenge that requires a strong understanding of how the mathematical choices that can be made influence the model, and how software abstractions can be made to flexibly allow a wide range of possibilities. Gaussian processes have seen some excellent, and highly used, software packages, such as GPML [Rasmussen and Nickisch, 2010], and GPy [The GPy authors, 2012–2014]. This thesis heavily used and contributed to GPflow [de G. Matthews et al., 2017], which uses TensorFlow [Abadi et al., 2015] for automatic differentiation. Automatic differentiation has greatly simplified the creation of GP models, as it did for deep learning models.

The future success of Bayesian deep learning models using GPs will similarly depend on the quality of the software available, and how well the mathematics is abstracted into software. Recent development in deep GPs [Salimbeni and Deisenroth, 2017]

provides hope that accurate inference methods can be conveniently expressed in code as well.

7.4 Conclusion

Bayesian non-parametric modelling in general, and Gaussian processes in particular, provide many advantages that modern machine learning methods can still benefit from. Computational drawbacks have been a large obstacle preventing the large-scale uptake of Gaussian processes, and widespread experimentation with different models. While there are still many open challenges, I hope that this thesis provides additional clarity which will help guide future work, and some useful models and tools that will prove to be useful in future solutions.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org. 74, 86, 125, 145

Ryan Prescott Adams, Iain Murray, and David J. C. MacKay. The Gaussian process density sampler. In *Advances in Neural Information Processing Systems 21*. 2009. 132, 133

Mauricio Alvarez, David Luengo, Michalis Titsias, and Neil D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010. 67, 68

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems 29*. 2016. 49

Matthew J Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby computational neuroscience unit, University College London, 2003. 8, 41

Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009. 91

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2016. 8, 37

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011. 8

Thang Bui, Daniel Hernandez-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep Gaussian processes for regression using approximate expectation propagation. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016. 141

Thang D. Bui, Josiah Yan, and Richard E. Turner. A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 18(104):1–72, 2017. 44, 45

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016. 91

François Chollet et al. Keras. https://keras.io, 2015. 145

Richard T Cox. Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13, 1946. 3

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural computation*, 14(3):641–668, 2002. 14, 45

Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep Gaussian processes. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 141

Andreas Damianou and Neil Lawrence. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 2013. 47, 62, 140, 141

Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research*, 17(42):1–62, 2016. 130

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *Journal of Machine Learning Research*, 2017. 86, 145

Marc P. Deisenroth and Carl E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, 2011. 20

Nicolas Durrande, David Ginsbourger, and Olivier Roustant. Additive covariance kernels for high-dimensional Gaussian process modeling. In *Annales de la Faculté de Sciences de Toulouse*, volume 21, 2012. 98

David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge, 2014. 91, 98

David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, 2013. 92, 116

David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive Gaussian processes. In *Advances in neural information processing systems*, 2011. 98, 116

Giancarlo Ferrari-Trecate, Christopher K. I. Williams, and Manfred Opper. Finite-dimensional approximation of Gaussian processes. In *Advances in Neural Information Processing Systems 11*. 1999. 25

R. Frigola. *Bayesian Time Series Learning with Gaussian Processes*. PhD thesis, University of Cambridge, 2015. 127, 135, 136, 138

Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In L. Bottou, C.J.C. Burges, Z. Ghahramani, M. Welling, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3156–3164. 2013. URL http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips26/1449.pdf. 136

Roger Frigola, Yutian Chen, and Carl Edward Rasmussen. Variational Gaussian process state-space models. In *Advances in Neural Information Processing Systems 27*. 2014. 60, 62, 134, 135, 136, 137, 163

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. 139

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 2016. 139

Yarin Gal and Richard Turner. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015. 88

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. In *Advances in Neural Information Processing Systems 29*. 2016. 92

Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984), 2012. 2

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521, 2015. 2

Mark Gibbs and David J. C. MacKay. *Efficient Implementation of Gaussian Processes*. 1997. 144

David Ginsbourger, Olivier Roustant, and Nicolas Durrande. Invariances of random fields paths, with applications in Gaussian process regression, 2013. 91, 98, 99

Roger Grosse and James Martens. A Kronecker-factored approximate Fisher matrix for convolution layers. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016. 89

Peter D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007. ISBN 0262072815. 2

James Hensman and Neil D. Lawrence. Nested variational compression in deep Gaussian processes, 2014. 128

James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence*, 2013. 43, 49, 56, 73, 145

James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems 28*. 2015a. 43, 112, 114

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015b. 43, 56

James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 2018. 39, 68

Daniel Hernández-Lobato and José Miguel Hernández-Lobato. Scalable Gaussian process classification via expectation propagation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2016. 44

Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 61

Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A distributed variational inference framework for unifying parallel sparse gaussian process regression models. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016. 61

Ferenc Huszár. Choice of recognition models in VAEs: a regularisation view. *inference. vc*, 2016. 89

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003. 2, 3

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014. 145

George S. Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, apr 1970. doi: 10.1214/aoms/1177697089. 14

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 111, 145

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2015. 139

J. Ko and D. Fox. Learning GP-BayesFilters via Gaussian process latent variable models. In *Proceedings of Robotics: Science and Systems*, 2009. doi: 10.15607/RSS.2009.V.029. 135

Risi Kondor. *Group theoretical methods in machine learning*. PhD thesis, Columbia University, 2008. 91, 98, 99

Karl Krauth, Edwin V. Bonilla, Kurt Cutajar, and Maurizio Filippone. AutoGP: Exploring the capabilities and limitations of Gaussian process models. 2016. 88, 114, 123

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html. 121

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012. 10

M. Kuss and C. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 2005. 44, 143

Miguel Lázaro-Gredilla. *Sparse Gaussian Processes for Large-Scale Machine Learning*. PhD thesis, Universidad Carlos III de Madrid, 2010. 70, 157

Miguel Lázaro-Gredilla and Anibal Figueiras-Vidal. Inter-domain gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, 2009. 26, 56, 67, 68, 69, 70, 71, 72, 77, 100, 157

Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 2010. 25, 28, 29, 35, 88

Quoc V. Le, Tamas Sarlos, and Alex Smola. Fastfood-computing hilbert space expansions in loglinear time. In *International Conference on Machine Learning*, 2013. 28

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. 101, 102

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521, 05 2015. 139

Yingzhen Li and Yarin Gal. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, 2017. 140

David J. C. MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992a. 139

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002. ISBN 0521642981. 2, 3, 4, 5, 6, 8, 93

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992b. 139

David JC MacKay. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998. 89, 97

David JC MacKay. Comparison of approximate methods for handling hyperparameters. *Neural computation*, 11(5):1035–1068, 1999. 8, 12

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems 27*. 2014. 102

Alexander G. de G. Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2016. 15, 43, 51, 54, 68, 100, 104, 106, 128

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 *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, 2016. 9, 41, 42, 127

Andrew McHutchon. *Nonlinear Modelling and Control using Gaussian Processes*. PhD thesis, University of Cambridge, 2014. 138

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 2006. 20

T. Minka, J.M. Winn, J.P. Guiver, S. Webster, Y. Zaykov, B. Yangel, A. Spengler, and J. Bronskill. Infer.NET 2.6, 2014. Microsoft Research Cambridge. http://research.microsoft.com/infernet. 44

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, 2001. 37, 44

Tom Minka and John Winn. Gates. In *Advances in Neural Information Processing Systems*, 2009. 131

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258. 2

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010. 89

Andrew Naish-Guzman and Sean Holden. The generalized FITC approximation. In *Advances in Neural Information Processing Systems 20*. 2008. 44, 47

Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1994. 123, 124, 139

H. Nickisch and CE. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 2008. 44, 143

Gaurav Pandey and Ambedkar Dukkipati. Learning by stretching deep networks. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1719–1727. JMLR Workshop and Conference Proceedings, 2014. URL http://jmlr.org/proceedings/papers/v32/pandey14.pdf. 102

Omiros Papaspiliopoulos and Gareth O. Roberts. Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models. *Biometrika*, 95(1):169–186, 2008. doi: 10.1093/biomet/asm086. 132

Yuan Qi, Ahmed H. Abdel-Gawad, and Thomas P. Minka. Sparse-posterior gaussian processes for general likelihoods. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010. 45

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005. 25, 31, 32, 33, 34, 35, 36, 60, 104

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*. 2008. 25, 28

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 2014. 41

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems 13*, 2001. 2, 19, 48, 93

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of machine learning research*, 11(Nov):3011–3015, 2010. 145

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. xiv, 6, 8, 9, 10, 12, 14, 16, 17, 20, 25, 28, 44, 55, 92, 115, 130, 136, 159, 161

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. 145

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 10

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017. 141, 145

Hilary L. Seal. Studies in the history of probability and statistics. XV: The historical development of the gauss linear model. *Biometrika*, 54(1/2):1, 1967. doi: 10.2307/2333849. 10

Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003. 9, 92

Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(02):69–106, Apr 2004. ISSN 1793-6462. doi: 10.1142/s0129065704001899. 9, 15

Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003. 30, 33, 35, 48

Ross D Shachter. Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998. 136

Edward Snelson. *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, University College London, 2007. 54, 64

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*. 2006. xiii, 30, 36, 47, 48, 49, 50, 67, 77, 104

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013. 140

The GPy authors. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, 2012–2014. 145

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv 1605.02688*, 2016. 145

M. Titsias. Variational inference for Gaussian and determinantal point processes. Advances in Variational Inference, NIPS 2014 Workshop, December 2014. URL http://www2.aueb.gr/users/mtitsias/papers/titsiasNipsVar14.pdf. 43

Michaelis K. Titsias. Variational model selection for sparse Gaussian process regression. Technical report, University of Manchester, 2009a. 48, 67, 68

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014. 41

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2009b. 25, 41, 42, 43, 44, 47, 48, 50, 51, 53, 54, 60, 61, 62, 63, 73, 74, 104, 117, 128

Michalis K. Titsias and Neil D. Lawrence. Bayesian Gaussian process latent variable model. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. 47, 62, 130, 131, 133

Felipe Tobar, Thang D Bui, and Richard E Turner. Learning stationary time series using gaussian processes with nonparametric kernels. In *Advances in Neural Information Processing Systems 28*. 2015. 68

Richard E. Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian Time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011. 49

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems 30*. 2017. 92

C. Walder, KI. Kim, and B. Schölkopf. Sparse multiscale Gaussian process regression. In *Proceedings of the 25th International Conference on Machine Learning*, 2008. 68

Jack Wang, Aaron Hertzmann, and David M. Blei. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems 18*. 2006. 135

Christopher K. I. Williams, Carl Edward Rasmussen, Anton Schwaighofer, and Volker Tresp. Observations on the nyström method for Gaussian processes. Technical report, 2002. 25

Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning*, 2013. 92

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, 2016. 91, 92

Appendix A

Inter-domain inducing features

A.1 Time-Frequency Inducing Features

The Time-Frequency Inducing Feature (TFIF) is obtained by shifting the Frequency Inducing Feature (FIF) integral projection [Lázaro-Gredilla and Figueiras-Vidal, 2009; Lázaro-Gredilla, 2010], giving:

$$\begin{aligned} g_{TFIF}(\mathbf{x}, \mathbf{z}) &= g_{FIF}(\mathbf{x} - \boldsymbol{\mu}, \mathbf{z}) \\ &= \prod_{d=1}^{D} \left(2\pi c_d^2\right)^{-\frac{1}{2}} \exp\left(-\sum_{d=1}^{D} \frac{(x_d - \mu_d)^2}{2c_d^2}\right) \cos\left(\omega_0 + \sum_{d=1}^{D} (x_d - \mu_d)\omega_d\right) \end{aligned} \quad \text{(A.1)}$$

Where $\mathbf{z} = \{\boldsymbol{\mu}, \boldsymbol{\omega}\}$. The expression for the covariance of the inducing variable $k_{TFIF}(\mathbf{z}, \mathbf{z}')$ given in Lázaro-Gredilla and Figueiras-Vidal [2009]; Lázaro-Gredilla [2010], however, is incorrect. Here we present the corrected expression, which turns out to differ only in a few terms from the expressions originally reported.

The covariance of interest is given by a double integration. The first integration simply returns the cross-covariance $k_{TFIF}(\mathbf{x}, \mathbf{z})$, which is correctly given in the earlier references. The second integral can be written as convolution of a Gaussian-cosine product with another Gaussian. This can be solved by using standard Fourier transform identities for convolutions and the spectra of Amplitude Modulated signals. We start from the expressions for the inducing output covariance, and express it in terms of a

convolution involving the cross-covariance of FIF.

$$\begin{aligned} k_{TFIF}(\mathbf{z}, \mathbf{z}') &= \iint k(\mathbf{x}, \mathbf{x}') g(\mathbf{x}, \mathbf{z}) g(\mathbf{x}', \mathbf{z}') \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{x}' \\ &= \int k_{TFIF}(\mathbf{x}, \mathbf{z}) g(\mathbf{x}, \mathbf{z}') \mathrm{d}\mathbf{x} = \int k_{FIF}(\mathbf{x} - \boldsymbol{\mu}, \boldsymbol{\omega}) g_{FIF}(\mathbf{x} - \boldsymbol{\mu}', \boldsymbol{\omega}') \mathrm{d}\mathbf{x} \\ &= \int k_{FIF}(\underbrace{\boldsymbol{\mu} - \boldsymbol{\mu}'}_{=\boldsymbol{\tau}} - \mathbf{x}, \boldsymbol{\omega}) g_{FIF}(\mathbf{x}, \boldsymbol{\omega}') \mathrm{d}\mathbf{x} \\ &= k_{FIF}(\boldsymbol{\tau}, \boldsymbol{\omega}) * g_{FIF}(\boldsymbol{\tau}, \boldsymbol{\omega}') \end{aligned} \tag{A.2}$$

$$\therefore \mathcal{F}\{k_{TFIF}(\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\omega}')\} = \mathcal{F}\{k_{FIF}(\boldsymbol{\tau}, \boldsymbol{\omega})\} \mathcal{F}\{g_{FIF}(\boldsymbol{\tau}, \boldsymbol{\omega})\} \tag{A.3}$$

In the Fourier domain, the convolution is a multiplication, solving the integral. The trick is to use the identity

$$\mathcal{F}\{\cos(f_0 t) s(t)\} = \frac{1}{2} S(f - f_0) + \frac{1}{2} S(f + f_0)\,, \tag{A.4}$$

to obtain the necessary Fourier transforms. Both will turn into a sum of two sums, symmetric about the origin. Multiplying these, we get a sum of 4 Gaussians. Using the same identity in the inverse, we obtain the result:

$$\begin{aligned} k_{TFIF}(\mathbf{z}, \mathbf{z}') = \frac{1}{2} \prod_{d=1}^{D} \left(\frac{l_d^2}{l_d^2 + c_d^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{c_d^2 l_d^2 (w_d + w_d'^2)}{2c_d^2 + l_d^2} \right) \exp\left(-\frac{1}{2} \frac{(\mu_d - \mu_d')^2}{2c_d^2 + l_d^2} \right) \\ \left[\exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{c_d^4 (\omega_d - \omega_d')^2}{2c_d^2 + l_d^2} \right) \cos\left(\sum_{d=1}^{D} \frac{c_d^2 (\omega_d + \omega_d')}{2c_d^2 + l_d^2} (\mu_d - \mu_d') - (\omega_0 - \omega_0') \right) + \right. \\ \left. \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{c_d^4 (\omega_d + \omega_d')^2}{2c_d^2 + l_d^2} \right) \cos\left(\sum_{d=1}^{D} \frac{c_d^2 (\omega_d - \omega_d')}{2c_d^2 + l_d^2} (\mu_d - \mu_d') - (\omega_0 + \omega_0') \right) \right] \end{aligned} \tag{A.5}$$

Appendix B

Inducing point updates for VFE and FITC

Here, we prove the claims of chapter 3 on how the objective functions for VFE and FITC change after adding an inducing input. From equation 3.1 we see that the only term that depends on the inducing inputs, is $Q_{\mathbf{ff}} = K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uf}}$. We can examine the effect of adding a new inducing input on the objective, by considering rank 1 updates to this matrix.

B.1 Adding a new inducing point

We begin with the covariance matrix for M inducing points $K_{\mathbf{uu}}$, and the corresponding approximate covariance $Q_{\mathbf{ff}}$. We denote versions of these matrices with a new inducing point added with a superscript $+$. We are interested in the updated $Q_{\mathbf{ff}}$:

$$Q_{\mathbf{ff}}^{+} = K_{\mathbf{fu}}^{+} (K_{\mathbf{uu}}^{+})^{-1} K_{\mathbf{uf}}^{+} \,. \tag{B.1}$$

To perform the rank 1 update, we simply perform a block inversion of $K_{\mathbf{uu}}^{+}$ [Rasmussen and Williams, 2005, p. 201]:

$$\begin{aligned} (K_{\mathbf{uu}}^{+})^{-1} &= \begin{pmatrix} K_{\mathbf{uu}} & k_{\mathbf{u}} \\ k_{\mathbf{u}}^{\intercal} & k \end{pmatrix}^{-1} \\ &= \begin{pmatrix} K_{\mathbf{uu}}^{-1} + \frac{1}{c}\mathbf{a}\mathbf{a}^{\intercal} & -\frac{1}{c}\mathbf{a} \\ -\frac{1}{c}\mathbf{a}^{\intercal} & \frac{1}{c} \end{pmatrix} \qquad\qquad \mathbf{a} = K_{\mathbf{uu}}^{-1} k_{\mathbf{u}} \end{aligned}$$

where $c = k - k_{\mathbf{u}}^{\intercal} K_{\mathbf{uu}}^{-1} k_{\mathbf{u}}$, $k_{\mathbf{u}}^{\intercal} = (k(\mathbf{z}_1, \mathbf{z}_{M+1}), \ldots, k(\mathbf{z}_M, \mathbf{z}_{M+1}))$ (the vector of covariances between the old and new inducing inputs), and $k = k(\mathbf{z}_{M+1}, \mathbf{z}_{M+1})$.

$$K_{\mathbf{uf}}^{+} = \begin{pmatrix} K_{\mathbf{uf}} \\ k_{\mathbf{f}}^{\intercal} \end{pmatrix}$$

where $k_{\mathbf{f}}^{\intercal} = (k(\mathbf{z}_{M+1}, \mathbf{x}_1), \ldots, k(\mathbf{z}_{M+1}, \mathbf{x}_N))$ is the vector of covariances between the data points and the new inducing input. $Q_{\mathbf{ff}}^{+}$ can now be found by a simple matrix product, resulting in a rank 1 update.

$$\begin{aligned}
Q_{\mathbf{ff}}^{+} &= K_{\mathbf{fu}}^{+} (K_{\mathbf{uu}}^{+})^{-1} K_{\mathbf{uf}}^{+} \\
&= K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uf}} + \frac{1}{c} (K_{\mathbf{fu}} \mathbf{a}\mathbf{a}^{\intercal} K_{\mathbf{uf}} + K_{\mathbf{fu}} \mathbf{a}\mathbf{a}^{\intercal} K_{\mathbf{uf}} \\
&\quad - K_{\mathbf{fu}} \mathbf{a} k_{\mathbf{f}}^{\intercal} - k_{\mathbf{f}} \mathbf{a}^{\intercal} K_{\mathbf{uf}} + k_{\mathbf{f}} k_{\mathbf{f}}^{\intercal}) \\
&= K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uf}} + \frac{1}{c} (K_{\mathbf{fu}} \mathbf{a} - k_{\mathbf{f}})(K_{\mathbf{fu}} \mathbf{a} - k_{\mathbf{f}})^{\intercal} \\
&= K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uf}} + \mathbf{b}\mathbf{b}^{\intercal} \\
&= Q_{\mathbf{ff}} + \mathbf{b}\mathbf{b}^{\intercal} \\
Q_{\mathbf{ff}}^{+} &= Q_{\mathbf{ff}} + \mathbf{b}\mathbf{b}^{\intercal}
\end{aligned}$$

We obtain $\mathbf{b} = \frac{1}{\sqrt{c}} (K_{\mathbf{fu}} K_{\mathbf{uu}}^{-1} k_{\mathbf{u}} - k_{\mathbf{f}})$ for the rank 1 update.

B.2 The VFE objective function always improves when adding an additional inducing input

The change in objective function can now be computed using $Q_{\mathbf{ff}}^{+}$.

$$\begin{aligned}
2(\mathcal{F}^{+} - \mathcal{F}) &= \log \left| Q_{\mathbf{ff}}^{+} + \sigma_n^2 I \right| - \log \left| Q_{\mathbf{ff}} + \sigma_n^2 I \right| + \mathbf{y}^{\intercal} (Q_{\mathbf{ff}}^{+} + \sigma_n^2 I)^{-1} \mathbf{y} - \mathbf{y}^{\intercal} (Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1} \mathbf{y} \\
&\quad + \frac{1}{\sigma_n^2} \operatorname{Tr}(K_{\mathbf{ff}} - Q_{\mathbf{ff}}^{+}) - \frac{1}{\sigma_n^2} \operatorname{Tr}(K_{\mathbf{ff}} - Q_{\mathbf{ff}}) \\
&= \log \left| Q_{\mathbf{ff}} + \mathbf{b}\mathbf{b}^{\intercal} + \sigma_n^2 I \right| - \log \left| Q_{\mathbf{ff}} + \sigma_n^2 I \right| \\
&\quad + \mathbf{y}^{\intercal} (Q_{\mathbf{ff}} + \mathbf{b}\mathbf{b}^{\intercal} + \sigma_n^2 I)^{-1} \mathbf{y} - \mathbf{y}^{\intercal} (Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{\sigma_n^2} \operatorname{Tr}(\mathbf{b}\mathbf{b}^{\intercal})
\end{aligned}$$

We extensively use the Woodbury identity [Rasmussen and Williams, 2005, p. 201] and to decompose $Q_{\mathbf{ff}}^{+}$ into $Q_{\mathbf{ff}}$ and its rank 1 update.

$$\begin{aligned}
2(\mathcal{F}^{+} - \mathcal{F}) &= \log(1 + \mathbf{b}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}) + \log\left|Q_{\mathbf{ff}} + \sigma_n^2 I\right| - \log\left|Q_{\mathbf{ff}} + \sigma_n^2 I\right| \\
&\quad + \mathbf{y}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{y} - \mathbf{y}^{\intercal}\frac{(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}\mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}}{1 + \mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}}\mathbf{y} \\
&\quad - \mathbf{y}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{y} + \frac{1}{\sigma_n^2}\operatorname{Tr}(\mathbf{b}\mathbf{b}^{\intercal}) \\
&= \log(1 + \mathbf{b}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}) - \frac{1}{\sigma_n^2}\operatorname{Tr}(\mathbf{b}\mathbf{b}^{\intercal}) \\
&\quad - \mathbf{y}^{\intercal}\frac{(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}\mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}}{1 + \mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}}\mathbf{y}
\end{aligned}$$

We can bound the first two terms by noting that

$$\begin{aligned}
\operatorname{Tr}(\mathbf{b}\mathbf{b}^{\intercal}) &= \mathbf{b}^{\intercal}\mathbf{b}\,, \\
\log(1 + x) &\leq x\,, \\
\mathbf{b}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b} &\leq \frac{1}{\sigma_n^2}\mathbf{b}^{\intercal}\mathbf{b}\,.
\end{aligned}$$

As a consequence, we have that the first two terms are bounded by zero:

$$\log(1 + \mathbf{b}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}) - \frac{1}{\sigma_n^2}\operatorname{Tr}(\mathbf{b}\mathbf{b}^{\intercal}) \leq 0\,.$$

The final term is bounded in the same way:

$$\begin{aligned}
-\mathbf{y}^{\intercal}\frac{(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}\mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}}{1 + \mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}}\mathbf{y} &= -\frac{(\mathbf{y}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b})^2}{1 + \mathbf{b}^{T}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b}} \\
&\leq -(\mathbf{y}^{\intercal}(Q_{\mathbf{ff}} + \sigma_n^2 I)^{-1}\mathbf{b})^2 \\
&\leq 0
\end{aligned}$$

Equalities hold when $\mathbf{f} = 0$, which is the case when an existing inducing point is added.

The FITC objective can increase or decrease with an added inducing point. While the complexity penalty seems to always improve by adding an inducing input, the data fit term can outweigh this benefit. We do not have a proof for the complexity penalty, but we hypothesise that it is true. The diagonal of $Q_{\mathbf{ff}} + G$ is always constrained to be the marginal GP variance. The correction term G is the heteroskedastic noise term.

As we add inducing points, the volume of $Q_{\mathbf{ff}}$ increases, reducing the amount of noise that is added. This improves the complexity penalty.

B.3 The heteroscedastic noise is decreased when new inducing inputs are added

While the objective function can change either way, the heteroscedastic noise, which is given by $\operatorname{diag}(K_{\mathbf{ff}} - Q_{\mathbf{ff}})$ always decreases or remains the same when a new inducing input is added:

$$\operatorname{diag}(K_{\mathbf{ff}} - Q_{\mathbf{ff}}^{+}) = \operatorname{diag}(K_{\mathbf{ff}} - (Q_{\mathbf{ff}} + \mathbf{b}\mathbf{b}^{\intercal})) \tag{B.2}$$

$$= \operatorname{diag}(K_{\mathbf{ff}} - Q_{\mathbf{ff}}) - \operatorname{diag}(\mathbf{b}\mathbf{b}^{\intercal}) \tag{B.3}$$

The diagonal elements of $\mathbf{b}\mathbf{b}^{\intercal}$ are given by b_m^2, which are always larger or equal to zero, such that the heteroscedastic noise always decreases (or stays the same).

Appendix C

Marginalisation of latent function in GPSSM

Here, we discuss the steps needed to marginalise out $f(\cdot)$ in the GPSSM to obtain the exact model formulated in Frigola et al. [2014]. The steps are straightforward, but confusing in notation, given common over-loadings of operators. Usually, when we refer to densities, we use the name of the random variable as both the identifier of the density, and as the location the density is evaluated at. I.e. $p([\mathbf{f}]_t|f(\cdot), x_{t-1})$ is the density of $[\mathbf{f}]_t$ evaluated at whatever value $[\mathbf{f}]_t$ takes. When this becomes ambiguous in the following, we will make this explicit by subscripting the density with its random variable, e.g. $p_{f(x_t)}(v)$. It should be clear that $p_{f(x_t)}(v)$ is the density function of the random variable $f(x_t)$ evaluated at v.

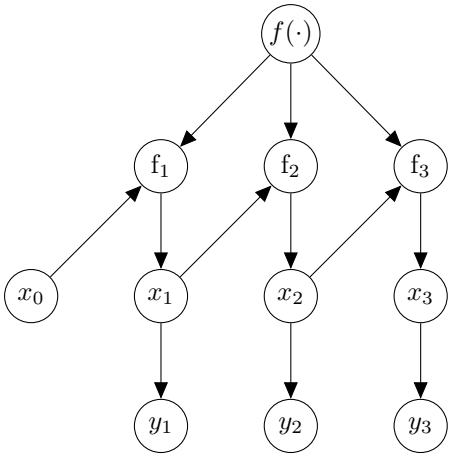


Fig. C.1 GPSSM with explicit evaluation variables.

We start with the formulation in figure 6.4, but we add the explicit evaluation variables as in figure 6.2.

$$p(\mathrm{f}_t|f(\cdot), x_{t-1}) = \delta(\mathrm{f}_t - f(x_{t-1}))\,. \tag{C.1}$$

To integrate out $f(\cdot)$, we have to solve the integral (we consider $\mathrm{f}_1 \dots \mathrm{f}_3$):

$$\begin{aligned}
p(\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3|x_0, x_1, x_2) &= \int \mathrm{d}f(x_{0:3}) p(f(x_{0:3})) \prod_{t=1} p(\mathrm{f}_t|f, x_{t-1}) \\
&= \int \mathrm{d}f(x_{1:3}) \Big(\int \mathrm{d}f(x_0) p(f(x_0)) \delta(\mathrm{f}_1 - f(x_0)) p(f(x_{1:3})|f(x_0)) \Big) \prod_{t=2}^{T} p(\mathrm{f}_t|f, x_{t-1}) \\
&= p_{f(x_0)}(\mathrm{f}_1) \int \mathrm{d}f(x_{2:3}) \Big(\int \mathrm{d}f(x_1) p(f(x_1)|f(x_0) = \mathrm{f}_1) \delta(\mathrm{f}_2 - f(x_1)) \\
&\qquad p(f(x_3)|f(x_0) = \mathrm{f}_1, f(x_1)) \Big) p(f_3|f, x_2) \\
&= p_{f(x_0)}(\mathrm{f}_1) p_{f(x_1)}(\mathrm{f}_2|f(x_0) = \mathrm{f}_1) \int \mathrm{d}f(x_3) p(f(x_3)|f(x_0) = \mathrm{f}_1, f(x_1) = \mathrm{f}_2, f(x_2)) \delta(\mathrm{f}_3 - f(x_2)) \\
&= p_{f(x_0)}(\mathrm{f}_1) p_{f(x_1)}(\mathrm{f}_2|f(x_0) = \mathrm{f}_1) p(\mathrm{f}_3|f(x_0) = \mathrm{f}_1, f(x_1) = \mathrm{f}_2)
\end{aligned} \tag{C.2}$$

The density function for f_n is simply a GP conditioned on $f_{1:n-1}$ at the inputs $x_{0:n-2}$. This structure of density function also shows the conditioning structure shown in figure 6.5.