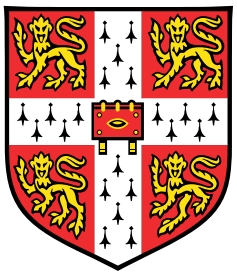
Efficiently-Parametrised Approximate Posteriors in Pseudo-Point Approximations to Gaussian Processes



Yuriko Kobe

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Hughes Hall College August 2021

To my parents Peter and Yasue. Your love, constant support and limitless patience are the greatest gifts I could wish for.

Acknowledgements

First and foremost, my immense gratitude to my supervisor, Dr. Richard E. Turner for his help and guidance, and importantly for his invaluable encouragement during this challenging pandemic year. I could not have accomplished this project or this MPhil degree without his mentorship. Furthermore, I would like to thank William Tebbutt whose initial ideas provided the basis for this project. I hugely appreciated our ongoing technical discussions, his insightful comments, and his overall support. I would also like to thank Dr Anthony Freeling and all the staff of Hughes Hall College, for their ongoing care during the pandemic. Finally, my heartfelt thanks to the ADTIS Language Centre for their English language skill coaching. I am truly grateful.

Declaration

I, Yuriko Kobe of Hughes Hall, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

I further declare the software that was used for this thesis. All computing experiments were carried out in Python. All experiments relied on standard libraries such as Numpy and TensorFlow. GPflow was modified and used for the Gaussian Process experiments. All plots were made using either the Matplotlib library or the TensorBoard tool. None of the above software except GPflow was modified and no other third-party software was used.

This dissertation contains fewer than 11,000 words including appendices and captions and has 10 figures.

Yuriko Kobe
August 2021

Abstract

Gaussian process models are flexible, robust to overfitting and give good estimates of predictive uncertainty. However, they are computationally expensive and approximations must be made in order to apply them to large datasets.

This project focuses on the pseudo-point approximation developed by Hensman et al. (2013), based on the foundations laid by Titsias (2009). Hensman's pseudo-point approximation is a powerful and efficient sparse variational inference method for Gaussian Processes with a stochastic optimization. This method is particularly beneficial when many observations have been made. Our aim is to improve this, first, by introducing an alternative parameterisation to the approximate posterior (prior + diagonal instead of dense matrix), whilst also utilising a natural gradient descent algorithm (Amari, 1998) (natural instead of standard gradients).

More precisely, at the pseudo-points, the precision matrix of the approximate posterior, originally parameterised as a dense matrix, is represented as the sum of prior precision and a positive-definite diagonal matrix. This is motivated by this parameterisation being optimal when not utilising pseudo-points (Opper and Archambeau, 2009). In addition, we extend the method by replacing a standard gradient descent algorithm with a natural gradient descent to optimise the variational parameters of the approximation.

We show the practical applicability and performance of these features on both synthetic and real datasets. This work additionally investigates the performance of the natural gradient descent algorithm in different settings (e.g. number of inducing inputs, number of training points, minibatch size, and noise variance), compared with a standard gradient descent algorithm, Adam (Kingma and Ba, 2014). These results give theoretical insight into the convergence of the natural gradients as well as the alternative paramterisation, and demonstrate the efficiency of the natural gradient algorithm.

Table of contents

List of figures

Nomenclature

Acronyms / Abbreviations

ELBO	Evidence Lower Bound
$\mathbf{F}_\theta$	Fisher information
GP	Gaussian Process
KL divergence	Kullback-Leibler divergence
M	the number of inducing inputs
NGD	Natural Gradient Descent
N	the number of training inputs
RBF	Radial-basis function kernel
SE kernel	Squared Exponential kernel
SVI	Stochastic Variational Inference
VFE	Variational Free Energy
VI	Variational Inference

Chapter 1

Introduction

1.1 Introduction and motivation

Gaussian Processes (GPs) (Rasmussen, 2003) are a powerful approach for inference on functions. By defining a distribution over functions, they provide a data-efficient and flexible inference method with reliable uncertainty estimates. However, the computational requirement of an exact implementation scales as $O(N^3)$ time, and as $O(N^2)$ memory, where N is the number of training points, resulting in computational intractability in many practical problems.

Fortunately, to address this, many approximation methods, so-called sparse GP approximation methods, have been developed in recent years (Bui et al., 2017; Csató and Opper, 2002; Hensman et al., 2013; Lawrence et al., 2003; Quinonero-Candela and Rasmussen, 2005; Seeger et al., 2003; Snelson and Ghahramani, 2006; Titsias, 2009). The key idea of those methods is to summarise the full GP via M inducing inputs (also known as pseudo-points or pseudo-inputs) where typically $M << N$. This allows us to retain the favourable properties of GPs but at a lower computational cost $O(NM^2)$ and memory storage $O(NM)$.

The main sparse GP approximation methods follow one of two frameworks (Bui et al., 2017): approximate generative models and approximate inference. The former can be interpreted as performing exact inference under an approximate GP prior (Quinonero-Candela and Rasmussen, 2005). While the latter can be seen as performing approximate (variational) inference under an exact GP prior. This project focuses on the latter framework, namely the sparse variational GP framework.

The variational free energy (VFE) method introduced by Titsias (2009) is perhaps the most well known approach in the sparse variational GP framework. By minimizing a Kullback-Leibler divergence (KL divergence) (Kullback and Leibler, 1951) between the approximating and posterior processes, it performs variational inference to optimize the

approximate posterior distribution. The main advantage of this method is that the inducing locations are variational parameters rather than model parameters, so they are protected from overfitting.

Based on the foundations laid in Titsias's VFE method, Hensman proposed a sparse variational GPs method for big data (Hensman et al., 2013). This uses exactly the same posterior approximation as that of Titsia, but finds the optimal variational parameters using a stochastic gradient descent algorithm. This allows us to reduce the computational complexity from $O(NM^2)$ to $O(M^3)$, meaning that we are free to increase the number of observations N.

This project develops Hensman's prior work, first, by introducing an alternative paramterisation, namely efficient $O(M)$ parameterisation, to the approximate posterior (prior + diagonal instead of dense matrix). In the prior work, the approximate posterior is parameterised by a mean vector $\mathbf{m}$ and a dense covariance matrix $\mathbf{S}$, earning $M + M(M+1)/2$ variational paramters.

$$\text{Original parameterisation:} \quad q(\mathbf{u}) := \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$$

Instead, we parameterise the precision matrix of the approximate posterior as the sum of prior precision ${K_{mm}}^{-1}$ and a positive-definite diagonal matrix $\mathbf{G}$ and the mean vector as the multiplication of the prior covariance and the mean in the affine transformation space. This formulation is motivated by the parameterisation being optimal when not utilising pseudo-points (Opper and Archambeau, 2009). It sacrifices the possibility of obtaining the optimal approximate posterior in favour of reducing the number of variational parameters from $M + M(M+1)/2$ to just $2M$.

$$\text{Alternative parameterisation:} \quad q(\mathbf{u}) := \mathcal{N}\left(\mathbf{u}; \mathbf{K_{mm}}\mu, \left[{\mathbf{K_{mm}}}^{-1} + \mathbf{G}\right]^{-1}\right)$$

In addition, we extend the prior work by replacing the standard gradients with the natural gradients in the stochastic optimization of the variational parameters (natural instead of standard gradients). Since the alternative variational parameters $[\mu, \text{diag}(\mathbf{G})]$ can be seen as natural parameters in the affine tranformation space, this alternative parameterisation is expected to accelerate, stabilise and make more robust the natural gradient descent algorithm.

1.2 Thesis outline and contributions

The remainder of this is thesis is organised as follows. In chapter 2 we present the necessary background for the rest of the thesis. We discuss the sparse variational GPs, the

natural gradient descent algorithm, and related work. In chapter 3 we propose an alternative parameterisation for the approximate posterior in sparse variational GPs, and present the motivation behind it. We demonstrate how natural gradients can be used to optimise the variational parameters of this approximation. In chapter 4 we describe the experimental methods and key concepts for the initialisation of parameters. In chapter 5 we first investigate a natural gradient descent compared to a standard gradient descent (Adam), by testing them in different settings (e.g. number of inducing inputs, number of training points, minibatch size, and noise variance). Then, we assess our new natural gradient algorithm with the $O(M)$ parameterisation, and propose potential improvements. We present results with a number of important findings:

- The key factors for NGD are M and the batch size
- If M is small enough, NGD is always faster than Adam
- If M is large and we do not use a minibatch, NGD is faster in the beginning, but both converge after the same number of iterations
- Only, if M is large and minibatch is used, NGD is slower, but it still reaches the same optimum as Adam
- $O(M)$ parameterisation of the approximate posterior in sparse variational GP methods can be efficiently combined with NGD using the affine transformation
- $O(M)$ parameterisation can lead to fast convergence of the stochastic natural gradient optimization

Lastly, conclusions, limitations of the proposed model, and comments over future research are made in Chapter 6.

Chapter 2

Background and Related Work

2.1 Gaussian Processes

A Gaussian Process (GP) (Rasmussen, 2003) is a stochastic process where each finite marginal distribution is a multivariate Gaussian distribution. It can be seen as a generalization of a multivariate Gaussian distribution to infinitely many variables. It is a powerful approach for inference on functions. By defining a distribution over functions, it allows direct inference and learning to take place in the function space, with reliable uncertainty estimates.

Consider a training dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{N \times D}$ is the set of training inputs and $\mathbf{y} = \{y_i\}_{i=1}^N \in \mathbb{R}^N$ is the corresponding target vector. Each entry y_i is a possible noisy and/or non-conjugate observation of the function $f_i \equiv f(\mathbf{x}_i) \in \mathbb{R}^N$ given by:

$$y_i \sim f(\mathbf{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}\left(0, \beta^{-1}\right) \tag{2.1}$$

We introduce a GP prior over the function of interest $f(\cdot)$. A GP is fully specified by its mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$ with the latter providing a way to evaluate the covariance between any two points in the function input space. For the sake of simplicity and without loss of generality, we will take the mean function $m(\mathbf{x})$ to be the zero function throughout this thesis.

$$f \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right) \tag{2.2}$$

Choosing the right kernel is critical for GPs and typically its form must be specified in advance as this restricts the property of the model and allows domain knowledge to be incorporated. Thus, we choose a kernel based on our prior knowledge on the model. In this project, we employ two common kernels: squared exponential (SE) and Matern $5/2$ kernel.

The SE kernel, also known as the Radial-basis function (RBF) kernel, is a stationary kernel and appropriate for modelling very smooth functions defined as follows:

$$k\left(\mathbf{x},\mathbf{x}'\right)=\sigma_f^2\exp\left(-\frac{1}{2}\frac{|\mathbf{x}-\mathbf{x}'|^2}{\ell^2}\right) \tag{2.3}$$

It is parameterised by its length scale $\ell>0$ and variance σ_f. The former describes how smooth the function of interest is, and the latter determines variation of function values. The Matern kernel can be seen as a generalization of the SE kernel with an additional parameter ν:

$$k\left(\mathbf{x},\mathbf{x}'\right)=\frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{\ell}\left|\mathbf{x}-\mathbf{x}'\right|\right)^{\nu}K_{\nu}\left(\frac{\sqrt{2\nu}}{\ell}\left|\mathbf{x}-\mathbf{x}'\right|\right) \tag{2.4}$$

where $K_\nu(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function. We use $\nu=\frac{2}{5}$ (i.e. Matern $5/2$ kernel) since it becomes a twice differentiable function.

Now we consider the distribution over the function values at the training points $\mathbf{X}$:

$$p(\mathbf{f})=\mathcal{N}\left(\mathbf{f}\mid\mathbf{0},\mathbf{K}_{nn}\right) \tag{2.5}$$

where $\mathbf{f}$ is a vector and $[\mathbf{K}_{nn}]_{ij}=k\left(\mathbf{x}_i,\mathbf{x}_j\right)$ is the covariance matrix. The conditional likelihood is also Gaussian:

$$p(\mathbf{y}\mid\mathbf{f})=\prod_{i=1}^{N}p\left(y_i\mid f_i\right)=\prod_{i=1}^{N}\mathcal{N}\left(y_i\mid f_i,\beta^{-1}\right) \tag{2.6}$$

Thus, following the definition, the marginal likelihood is given by:

$$p(\mathbf{y})=\int p(\mathbf{y}\mid\mathbf{f})p(\mathbf{f})d\mathbf{f}=\mathcal{N}\left(\mathbf{y}\mid\mathbf{0},K'_{nn}\right) \tag{2.7}$$

where $K'_{nn}=\mathbf{K}_{nn}+\beta^{-1}I$. The predictive posterior distribution at test point $\mathbf{X}_*$ is given by:

$$\begin{aligned}p(\mathbf{f}_*\mid\mathbf{X}_*,\mathcal{D})&=\int p(\mathbf{f}_*\mid\mathbf{X}_*,\mathbf{f})p(\mathbf{f}\mid\mathcal{D})d\mathbf{f}\\&=\mathcal{N}\left(\mathbf{f}_*\mid\mathbf{K}_{*n}\left(K'_{nn}\right)^{-1}\mathbf{y},\mathbf{K}_{**}-\mathbf{K}_{*n}\left(K'_{nn}\right)^{-1}\mathbf{K}_{*n}^{\top}\right)\end{aligned} \tag{2.8}$$

Despite simple closed Gaussian form, the computation of both the marginal likelihood and the predictive posterior require a matrix inversion operation to be performed, meaning that the computational cost scales as $O(N^3)$ time and $O(N^2)$ storage. Thus, good approximations are essential for practical inference in GPs.

2.2 Sparse variational Gaussian Processes

2.2.1 Variational Inference in sparse Gaussian Processes

Sparse GP approximations have been developed to address the computational cost issue of the original GPs. In sparse GP methods, a low rank approximation is built to the covariance matrix based around M inducing points (also known as pseudo-points) where the number of inducing points M is a user selected parameter and typically $M << N$. This approximation allows us to avoid the potential redundancy of the data - commonly more data is observed than is actually necessary to represent the posterior distribution - leading a computational cost of $O(NM^2)$ and memory of $O(NM)$.

Combining the idea of Variational Inference (VI) together with sparse GP approximation, sparse variational GP methods are introduced. In general, VI methods aim to estimate an intractable distribution. They define a parameterised tractable distribution, namely a variational distribution $q(\mathbf{f})$, and try to minimize the *distance* between the approximation and the true distribution (Jordan et al., 1999). In the context of sparse GP approximation, $q(\mathbf{f})$ is used to approximate a true posterior distribution over M inducing points. This approximation is made as close as possible to the true posterior by minimizing the Kullback-Leibler divergence (Kullback and Leibler, 1951)).

$$q^*(\mathbf{f}) = \arg\min_q \mathrm{KL}(q(\mathbf{f})||p(\mathbf{f} \mid \mathbf{y}, \theta))$$

where θ is the hyperparameters. The KL term can be re-written using the predictive likelihood and the Evidence Lower Bound (ELBO):

$$\begin{aligned}
\mathrm{KL}(q(\mathbf{f})||p(\mathbf{f} \mid \mathbf{y}, \theta)) &:= \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f} \mid \mathbf{y}, \theta)} df \\
&= E_{q(\mathbf{f})}[\log q(\mathbf{f})] - E_{q(\mathbf{f})}\left[\log \frac{p(\mathbf{y}, \mathbf{f} \mid \theta)}{p(\mathbf{y} \mid \theta)}\right] \\
&= E_{q(\mathbf{f})}[\log q(\mathbf{z})] - E_{q(\mathbf{f})}[\log p(\mathbf{y}, \mathbf{z} \mid \theta)] + E_{q(\mathbf{f})}[\log p(\mathbf{y} \mid \theta)] \\
&= \underbrace{\log p(\mathbf{y} \mid \theta)}_{\text{Predictive Likelihood}} \underbrace{- E_{q(\mathbf{f})}\left[\log \frac{p(\mathbf{y}, \mathbf{f} \mid \theta)}{q(\mathbf{f})}\right]}_{\text{ELBO}}
\end{aligned}$$

Re-arranging, this expression yields:

$$\underbrace{E_{q_\lambda(\mathbf{f})}\left[\log \frac{p(\mathbf{y},\mathbf{f}\mid\theta)}{q_\lambda(\mathbf{f})}\right]}_{\text{ELBO}} = \underbrace{\log p(\mathbf{y}\mid\theta)}_{\text{Predictive Likelihood}} - \mathrm{KL}(q_\lambda(\mathbf{f})||p(\mathbf{f}\mid\mathbf{y},\theta)) \tag{2.9}$$

Thus, minimizing the KL divergence is equivalent to maximizing the ELBO. In the context of sparse GP approximation, the log-predictive distribution itself is intractable. Thus, using the ELBO as a loss function can be a useful alternative.

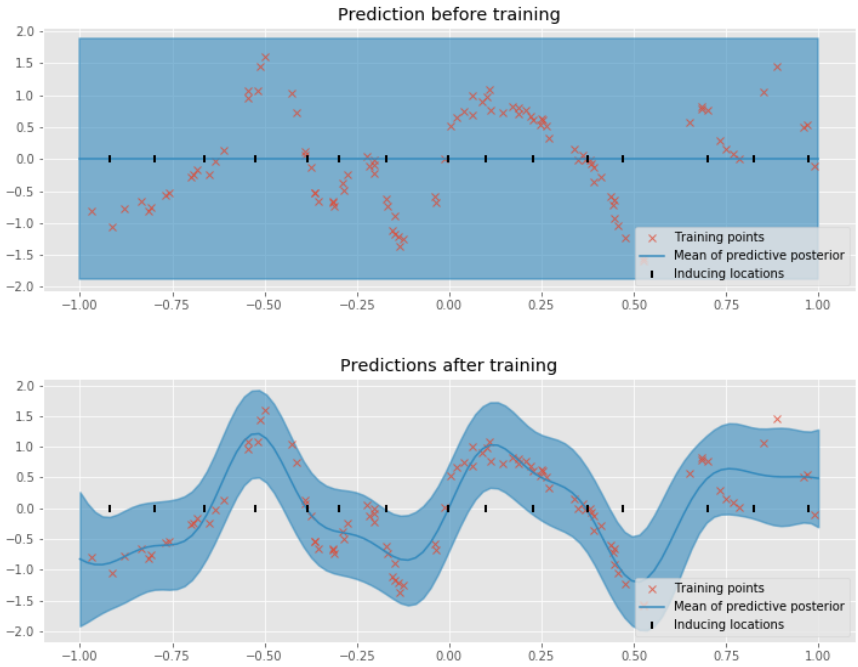


Fig. 2.1 The prediction samples before training and after training in the case of $N = 100, M = 15$, and the observation noise variance 0.01. The dataset is created from a simple sinusoidal function.

Figure 2.1 show sample outputs from a sparse variational GP model before and after training with 100 training inputs and 15 inducing inputs. We can see that it effectively learns the underlying function using a small number of inducing inputs while estimating the reliable uncertainty.

2.2.2 Variational Free Energy (VFE)

The variational free energy (VFE) method introduced by Titsias (2009) is perhaps one of the most well known approaches in the sparse variational GP framework. By minimizing a

KL divergence between the approximating and posterior processes, it performs variational inference to optimize the approximate posterior distribution.

The key idea of the sparse variational GP is augmentation. Let us assume the same setting as in the previous sections. VFE augments the initial joint distribution $p(\mathbf{y},\mathbf{f})$ with an additional latent vector $\mathbf{u} \in \mathbb{R}^N$ which contains values of the function f at the inducing locations $\mathbf{Z} \in \mathbb{R}^{M\times D}$. Thus, the augmented joint distribution becomes:

$$p(\mathbf{y},\mathbf{f}) = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u}). \tag{2.10}$$

where $p(\mathbf{u})$ is the marginal GP prior over $\mathbf{u}$, and $p(\mathbf{f} \mid \mathbf{u})$ is the conditional GP prior. Each terms on the right hand side of the above equation can be expressed as a Gaussian distribution as follows:

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{mm}) \tag{2.11}$$

$$p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}, \tilde{K}\right) \tag{2.12}$$

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \beta^{-1}I\right) \tag{2.13}$$

where $\mathbf{K}_{mm}$ is a $M \times M$ prior covariance matrix obtained by evaluating the kernel function at $\mathbf{Z}$, where $\mathbf{K}_{mn}$ and $\mathbf{K}_{nm} = \mathbf{K}_{mn}^\top$ are cross covariance matrices obtained by evaluating the kernel function between $\mathbf{Z}$ and $\mathbf{X}$, and where $\tilde{K} = \mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$.

Note that the inducing inputs $\mathbf{Z}$, which lie in the same space as the training inputs $\mathbf{X}$, play the role of variational parameters rather than model parameters. This means that they can be optimized to improve the approximation, and also they are protected from overfitting.

By applying Jensen's inequality, we obtain a lower bound to the exact log-marginal likelihood of the model:

$$\begin{aligned}
\log p(\mathbf{y} \mid \theta) &= \log \int p(\mathbf{y},\mathbf{f} \mid \theta) d\mathbf{f} \\
&\geq \int q(\mathbf{f}) \log \frac{p(\mathbf{y},\mathbf{f} \mid \theta)}{q(\mathbf{f})} d\mathbf{f} \\
&= \mathbb{E}_{q(\mathbf{f})} \log \frac{p(\mathbf{y},\mathbf{f} \mid \theta)}{q(\mathbf{f})} \\
&= \mathcal{L}_{VFE}
\end{aligned} \tag{2.14}$$

where θ represents the hyperparameters. The difference between the lower bound $\mathcal{L}_{VFE}$ and the exact log-marginal likelihood can be expressed by the KL divergence between the

variational distribution and the true posterior:

$$\log p(\mathbf{y} \mid \theta) - \mathcal{L}_{VFE} = \mathrm{KL}[q(\mathbf{f}) \| p(\mathbf{f} \mid \mathbf{y}, \theta)] \tag{2.15}$$

Therefore, variational inference is performed by explicitly minimizing the distance between a variational distribution and the true posterior distribution which is equivalent to maximizing the lower bound $\mathcal{L}_{VFE}$. Note that this result agrees with the VI equation 2.9. Also note that the exact marginal likelihood is recovered only when $q(\mathbf{f}) = p(\mathbf{f} \mid \mathbf{y}, \theta)$. The variational distribution is chosen to be the form:

$$q(\mathbf{f}) = q(\mathbf{f}_{\neq \mathbf{u}}, \mathbf{f} \mid \theta) = p(\mathbf{f}_{\neq \mathbf{u}} \mid \mathbf{u}, \theta) q(\mathbf{u}) \tag{2.16}$$

where $\mathbf{f}_{\neq \mathbf{u}}$ is all of the elements of f not in $\mathbf{u}$, $p(\mathbf{f} \mid \mathbf{u})$ is the conditional GP prior that appears also in the joint equation 2.12, and $q(\mathbf{u})$ is a variational distribution over the inducing variables. This, with some linear algebraic manipulation, leads to

$$\mathcal{L}_{VFE} = \log \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} + \beta^{-1}\mathbf{I}\right) - \frac{1}{2}\beta \operatorname{tr}(\widetilde{\mathbf{K}}) \tag{2.17}$$

with the implicit approximating distribution $q(\mathbf{u})$ having precision

$$\Lambda = \beta \mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1} + \mathbf{K}_{mm}^{-1} \tag{2.18}$$

and mean

$$\hat{\mathbf{u}} = \beta \Lambda^{-1}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}\mathbf{y}. \tag{2.19}$$

2.2.3 Sparse variational Gaussian Processes for big data

Based on the foundations laid in Titsias's VFE method, Hensman proposed a sparse variational GPs method for big data (Hensman et al., 2013). It uses exactly the same posterior approximation as that of Titsia, but finds the optimal variational parameters using a stochastic variational inference (SVI) algorithm so that the complexity per optimization step is reduced from $O\left(NM^2\right)$ to $O\left(M^3\right)$. This reduction in computational costs allows us to increase the number of observations N.

We start with the same setting as VFE. By defining a latent vector $\mathbf{u} \in \mathbb{R}^N$ which contains values of the function f at the inducing locations $\mathbf{Z} \in \mathbb{R}^{M \times D}$, equations 2.10 to 2.13 remain the same as before. Then, by applying Jensen's inequality, we obtain a lower bound to the

conditional probability:

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{u}) &= \log \int p(\mathbf{y}, \mathbf{f} \mid \mathbf{u}) d\mathbf{f} \\
&= \log \int p(\mathbf{f} \mid \mathbf{u}) p(\mathbf{y} \mid \mathbf{f}) d\mathbf{f} \\
&\geq \int p(\mathbf{f} \mid \mathbf{u}) \log p(\mathbf{y} \mid \mathbf{f}) d\mathbf{f} \\
&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})} \log p(\mathbf{y} \mid \mathbf{f}) \\
&\triangleq \mathcal{L}_1
\end{aligned} \tag{2.20}$$

This lower bound can be calculated with computational cost of $O(M^3)$. Using factorisation across the data $p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^{n} p(y_i \mid f_i)$ and equations 2.12 and 2.13, we can rewrite this lower bound as follows:

$$\exp(\mathcal{L}_1) = \prod_{i=1}^{n} \mathcal{N}\left(y_i \mid \mu_i, \beta^{-1}\right) \exp\left(-\frac{1}{2}\beta \tilde{k}_{i,i}\right) \tag{2.21}$$

where $\mu = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}$ and $\tilde{k}_{i,i}$ is the i^{th} diagonal element of $\widetilde{\mathbf{K}}$. As a comparison to equation 2.15, the difference between this lower bound $\mathcal{L}_1$ and the true log conditional likelihood is derived from the KL divergence between the posterior given both the inducing inputs and training data $p(\mathbf{f} \mid \mathbf{u}, \mathbf{y})$ and the posterior given only inducing inputs $p(\mathbf{f} \mid \mathbf{u})$:

$$\log p(\mathbf{y} \mid \mathbf{u}) - \mathcal{L}_1 = KL[p(\mathbf{f} \mid \mathbf{u}) \| p(\mathbf{f} \mid \mathbf{u}, \mathbf{y})] \tag{2.22}$$

This formula means that, if $M = N$ and the inducing inputs are located at the same place as the training inputs, $u = f$, $\mathbf{K}_{mm} = \mathbf{K}_{nm} = \mathbf{K}_{nn}$, and $\widetilde{\mathbf{K}} = 0$ are held and we can recover the true conditional likelihood $\exp(\mathcal{L}_1) = p(\mathbf{u} \mid \mathbf{f})$. Therefore, maximizing the lower bound with respect to variational parameters $\mathbf{Z}$ is equivalent to maximizing this KL divergence and ensures that $\mathbf{Z}$ are distributed amongst the training data $\mathbf{X}$.

Note that substituting the standard expression of this lower bound $\exp(\mathcal{L}_1)$ for the conditional likelihood $p(\mathbf{y} \mid \mathbf{u})$ gives a tractable bound on the marginal likelihood which is

exactly the same as that of the VFE method (see equation 2.14):

$$\begin{aligned}\log p(\mathbf{y}) &= \log \int p(\mathbf{y},\mathbf{u}) d\mathbf{u} \\ &= \log \int p(\mathbf{y} \mid \mathbf{u}) p(\mathbf{u}) d\mathbf{u} \\ &\geq \log \int \exp(\mathcal{L}_1) p(\mathbf{u}) d\mathbf{u} \\ &= \mathcal{L}_{VFE}\end{aligned} \tag{2.23}$$

Stochastic variational inference (SVI)

Given the expression for the lower bound $\mathcal{L}_1$, the fundamental problem for inference is to maximize the lower bound. To tackle this issue, an approach, called stochastic variational inference (SVI), is used. It allows variational inference for very large datasets. Although SVI is a powerful approach, in order to use it in a sparse GP model, some modification on the lower bound is required. In the following, we derive a new lower bound $\mathcal{L}_2$ which includes an explicit variational distribution $q(u)$, enabling SVI.

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) &\geq \mathcal{L}_{VFE} \\ &= \log \int \exp(\mathcal{L}_1) p(\mathbf{u}) d\mathbf{u} \\ &= \log \int \exp(\mathcal{L}_1) p(\mathbf{u}) \frac{q(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u} \\ &\geq \int (\mathcal{L}_1 + \log p(\mathbf{u}) - \log q(\mathbf{u}))\, q(\mathbf{u}) d\mathbf{u} \\ &= \mathbb{E}_{q(\mathbf{u})} (\mathcal{L}_1 + \log p(\mathbf{u}) - \log q(\mathbf{u})) \\ &\triangleq \mathcal{L}_2\end{aligned} \tag{2.24}$$

From the above we know that the optimal distribution is Gaussian, so we parameterise it as $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S})$. The bound $\mathcal{L}_2$ becomes:

$$\mathcal{L}_2 = \sum_{i=1}^{N} \left\{ \log \mathcal{N}\left(y_i \mid \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{m}, \beta^{-1}\right) - \frac{1}{2}\beta \widetilde{k}_{i,i} - \frac{1}{2} \operatorname{tr}(\mathbf{S}\mathbf{\Lambda}_i) \right\} - \mathrm{KL}(q(\mathbf{u}) || p(\mathbf{u})) \tag{2.25}$$

where $\mathbf{k}_i$ is a vector of the i^{th} column of $\mathbf{K}_{mn}$ and $\mathbf{\Lambda}_i = \beta \mathbf{K}_{mm}^{-1} \mathbf{k}_i \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1}$. The gradients of $\mathcal{L}_2$ with respect to the parameters of $q(\mathbf{u})$ are computed as:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{m}} = \beta \mathbf{K}_{mm}^{-1} \mathbf{K}_{mn} \mathbf{y} - \mathbf{\Lambda} \mathbf{m} \tag{2.26}$$

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{S}} = \frac{1}{2} \mathbf{S}^{-1} - \frac{1}{2} \mathbf{\Lambda} \tag{2.27}$$

Setting the derivatives to zero recovers the optimal solution $\mathbf{\Lambda}^{-1} = \mathbf{S}$, $\hat{\mathbf{u}} = \mathbf{m}$. The equality in $\mathcal{L}_{VFE} \geq \mathcal{L}_2$ holds only at this unique maximum. Now, the simplest approach would be stochastic gradient ascent which takes steps in the direction given by the gradient with respect to both the mean and covariance of the approximate posterior $\mathbf{m}, \mathbf{S}$. However, this approach would not guarantee the symmetry or positive definiteness of the covariance matrix $\mathbf{S}$, meaning that it requires further constraints. To address this issue, a scalable algorithm for approximating posterior distributions is suggested by Hoffman et al. (2013). Instead of the standard gradient, it uses the natural gradient (Amari, 1998) which ensures the positive definiteness of the covariance matrix while reducing the computational cost.

Here we briefly introduce the stochastic natural gradient algorithm used in (Hensman et al., 2013). See section 2.3 for detailed explanation on the natural gradient descent. The approximate natural gradient is the usual gradient re-scaled by the inverse Fisher information: $\tilde{\nabla}_\theta \mathcal{L} = (\nabla_\theta \mathcal{L}) \mathbf{F}_\theta^{-1}$. To work with the natural gradients of the distribution $q(\mathbf{u})$, we first recall the natural parameters θ and expectation parameters η :

$$\theta_1 = \mathbf{S}^{-1}\mathbf{m}, \quad \theta_2 = -\frac{1}{2}\mathbf{S}^{-1}$$
$$\eta_1 = \mathbf{m}, \quad \eta_2 = \mathbf{m}\mathbf{m}^\top + \mathbf{S}$$

In the exponential family, the Fisher information takes a particularly simple form in the natural parameters (Hensman et al., 2012). Using this simple form, the natural gradient can be written as the standard gradient with respect to the expectation parameters:

$$\tilde{\nabla}_\theta \mathcal{L} = (\nabla_\theta \mathcal{L}) \mathbf{F}_\theta^{-1} = \frac{\partial \mathcal{L}}{\partial \theta} (\frac{\partial \theta}{\partial \eta})^{-1} = \frac{\partial \mathcal{L}}{\partial \eta}.$$

Using $\theta_{(t+1)} = \theta_{(t)} + \gamma \tilde{\nabla}_\theta \mathcal{L}$ where γ is the step size, the natural gradient update can be expressed as:

$$\begin{aligned}
\theta_{2(t+1)} &= -\frac{1}{2}\mathbf{S}_{(t+1)}^{-1} \\
&= -\frac{1}{2}\mathbf{S}_{(t)}^{-1} + \ell\left(-\frac{1}{2}\mathbf{\Lambda} + \frac{1}{2}\mathbf{S}_{(t)}^{-1}\right), \\
\theta_{1(t+1)} &= \mathbf{S}_{(t+1)}^{-1}\mathbf{m}_{(t+1)} \\
&= \mathbf{S}_{(t)}^{-1}\mathbf{m}_{(t)} + \ell\left(\beta\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}\mathbf{y} - \mathbf{S}_{(t)}^{-1}\mathbf{m}_{(t)}\right)
\end{aligned}$$

and taking a step of unit length then recovers the same solution as described above in equation 2.26. This confirms the result discussed by Hensman et al. (2012) that taking this unit step is the same as performing a VB update. We can now obtain stochastic approximations to the natural gradient by considering the data either individually or in minibatches. We note the convenient result that the natural gradient for θ_2 is positive definite (i.e. $\mathbf{\Lambda} = \mathbf{K}_{mm}^{-1} + \sum_i \Lambda_i\big)$. This means that taking a step in the natural gradient direction always leads to a positive definite matrix, and our implementation need not parameterise $\mathbf{S}$ in any way to ensure positive definiteness, $cf.$ standard gradient approaches on covariance matrices.

Note that sparse variational approximate inference in non-conjugate settings has been addressed in (Hensman et al., 2015).

2.3 Natural Gradient Descent

Natural gradient was used for variational inference for the first time by Sato (2001), where it was shown that for an exponential family conditionally conjugate model, the Natural Gradient Descent (NGD) corresponds exactly to the fixed point variational update as long as the step size is set to 1. In conjugate cases, the simplification of the update steps of the natural gradient is investigated by Hensman et al. (2012); Hoffman et al. (2013) achieving fast convergence. Recent works extended the use of NGD to the non-conjugate cases achieving improved speed of convergence when compared to the standard gradient approach (Khan and Nielsen, 2018; Salimbeni et al., 2018). The three most important advantages of natural gradients are,

1. Paths following the natural gradient are invariant to reparameterisation (Martens, 2014),
2. The natural gradient direction is given by the ordinal gradient rescaled by the inverse Fisher information matrix $\tilde{\nabla}_\xi \mathcal{L} = (\nabla_\xi \mathcal{L})\mathbf{F}_\xi^{-1}$ (Amari, 1998),
3. NGD can solve problems in ill-conditioned settings where standard gradient methods fail (Salimbeni et al., 2018).

Let ξ be the parameters of a distribution p_ξ, and $\mathbf{F}(\xi)$ the corresponding Fisher information matrix. Amari (1998) proposed Natural Gradient Ascent, which is a form of pre-conditioned Gradient Ascent in a function f of ξ :

$$\xi^{(t+1)} = \xi^{(t)} + l^{(t+1)}\mathbf{F}\left(\xi^{(t)}\right)^{-1} \frac{\mathrm{d}f}{\mathrm{d}\xi}\bigg|_{\xi:=\xi^{(t)}}$$

for step sizes $l^{(t+1)} > 0$. Note the difference between Gradient Ascent and Gradient Descent. Essentially they are the same method to calculate the slope of objective function, but Gradient Ascent aims at maximizing the objective function as above, while Gradient Descent aims at minimizing some objective function. Thus, NGD updates can be expressed as:

$$\xi^{(t+1)} = \xi^{(t)} - l^{(t+1)}\mathbf{F}\left(\xi^{(t)}\right)^{-1} \frac{\mathrm{d}f}{\mathrm{d}\xi}\bigg|_{\xi:=\xi^{(t)}}$$

In other words, if the objective function is *convex* we use Gradient Descent and if it is *concave* we use we use Gradient Ascent. In practice we can easily switch between the two by setting the loss function negative $-\mathcal{L}$.

2.3.1 Conjugacy in Exponential Families

Consider an exponential family prior distribution. Following the definition of the exponential family, it can be written as:

$$p(\mathbf{u}) := h(\mathbf{u}) \exp\left(\langle \theta_0, \phi(\mathbf{u}) \rangle - A(\theta_0)\right) \tag{2.28}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, $\theta_0 \in \mathbb{R}^D$ is the natural parameters, $h : \mathbb{R}^P \to \mathbb{R}$ is the base measure, $\phi : \mathbb{R}^P \to \mathbb{R}^D$ is the sufficient statistic function, and $A : \mathbb{R}^D \to \mathbb{R}$ is the log partition function. For example, the natural parameters of a Gaussian with mean $\mathbf{m} \in \mathbb{R}^D$ and covariance matrix $\mathbf{S} \in \mathbb{R}^{D \times D}$ are given by stacking $\mathbf{S}^{-1}\mathbf{m}$ and $\mathrm{vec}\left(-\frac{1}{2}\mathbf{S}^{-1}\right)$ into a single vector, where $\mathrm{vec}(\cdot)$ transforms a matrix into a vector by stacking its columns on top of one another.

$$\text{natural parameters: } \theta = [\theta_1, \theta_2] = \left[\mathbf{S}^{-1}\mathbf{m}, \mathrm{vec}\left(-\frac{1}{2}\mathbf{S}^{-1}\right)\right]$$

This prior has a special relationship to the likelihood:

$$p(\mathbf{y} \mid \mathbf{u}) := S \exp(\langle \tilde{\theta}, \phi(\mathbf{u}) \rangle) \tag{2.29}$$

Furthermore, the corresponding posterior has the same form as the prior:

$$p(\mathbf{u} \mid \mathbf{y}) = h(\mathbf{u}) \exp\left(\langle \theta_1, \phi(\mathbf{u}) \rangle - A(\theta_1)\right), \quad \theta_1 := \theta_0 + \tilde{\theta} \ . \tag{2.30}$$

This is known as a *conjugacy* relationship between the prior and likelihood, and we say that *the likelihood is conjugate to the prior*.

Letting p_θ be some exponential family, written in terms of its natural parameters θ. We can equivalently represent it through its expectation parameters, which are defined as:

$$\eta(\theta) := \mathrm{E}_{p_\theta}[\phi(\mathbf{u})] \tag{2.31}$$

For example, for a Gaussian, these are given by stacking $\mathbf{m}$ and $\mathrm{vec}\left(\mathbf{m}\mathbf{m}^\top + \mathbf{C}\right)$ into a single vector.

$$\text{expectation parameters: } \eta = [\eta_1, \eta_2] = \left[\mathbf{m}, \mathbf{m}\mathbf{m}^\top + \mathbf{S}\right]$$

The expectation parameters have two important properties. Firstly, they are equal to the gradient of the log partition function with respect to the natural parameters:

$$\eta = \frac{\mathrm{d}A}{\mathrm{d}\theta} \tag{2.32}$$

Thus, manipulating equation 2.31 and 2.32 yields:

$$\eta(\theta) = \mathbb{E}_{p_\theta}[\phi(\mathbf{u})] = \left.\frac{\mathrm{d}A}{\mathrm{d}\theta}\right|_{\theta:=\theta_t} \tag{2.33}$$

Secondly, the Fisher information matrix for an exponential family distribution p_θ is

$$\mathbf{F}_\theta = \frac{\mathrm{d}\mu}{\mathrm{d}\eta} \tag{2.34}$$

2.3.2 Natural gradients in variational models

Given the expression for the lower bound $\mathcal{L}$, the fundamental problem is to minimize the KL divergence between the approximate variational distribution and the true distribution. The simplest approach would be to optimize the objective using gradient descent which takes steps in the direction given by the standard gradient. However, for probability distribution, the standard gradient may derive an unnatural direction since it is based on Euclidean distances. Consider the two pairs of Gaussians $\mathcal{N}(0,0.1),\mathcal{N}(1,0.1)$ and $\mathcal{N}(0,1000.1),\mathcal{N}(1,1000.1)$. Whilst the former pair are different, and the latter pair almost identical, the Euclidean distance is the same. To address this issue, recent works use a NGD method which takes steps in the direction given by the natural gradients instead of the ordinal gradients (Hensman et al., 2012; Hoffman et al., 2013).

Consider performing variational inference in a model whose prior has the form of equation 2.28, but whose likelihood is arbitrary. We will constrain our approximate variational posterior be of the same form as the prior:

$$q_{\theta_q}(\mathbf{u}) := h(\mathbf{u}) \exp\left(\left\langle \theta_q, \phi(\mathbf{u}) \right\rangle - A\left(\theta_q\right)\right) \tag{2.35}$$

Then, a natural gradient step is:

$$\theta_q^{(t+1)} = \theta^{(t)} + l^{(t+1)} \left[\mathbf{F}^{(t)}\right]^{-1} \left.\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_q}\right|_{\theta_q=\theta_q^{(t)}} \tag{2.36}$$

where $\mathcal{L}$ is the ELBO, which for the time being we will treat as being just a function of θ_q, and where $\mathbf{F}^{(t)} := \mathbf{F}\left(\theta_q^{(t)}\right)$. It turns out that the consequence of the assumptions that we

have made about the prior and approximate posterior simplify $\mathcal{L}$ substantially. To see this, first write out $\mathcal{L}$ using the expressions for the prior and q_{θ_q}:

$$\begin{aligned}\mathcal{L}\left(\theta_q\right) &= \mathbb{E}_{q_{\theta_q}}\left[\log \frac{p(\mathbf{y} \mid \mathbf{u})p(\mathbf{u})}{q_{\theta_q}(\mathbf{u})}\right] \\ &= \mathbb{E}_{q_{\theta_q}}[\log p(\mathbf{y} \mid \mathbf{u})] + \mathbb{E}_{q_{\theta_q}}\left[\left(\theta_0 - \theta_q\right)^\top \phi(\mathbf{u}) + A\left(\theta_q\right) - A\left(\theta_0\right)\right] \\ &= r\left(\theta_q\right) + \left(\theta_0 - \theta_q\right)^\top \eta\left(\theta_q\right) + A\left(\theta_q\right) - A\left(\theta_0\right), \end{aligned} \tag{2.37}$$

where $r\left(\theta_q\right) := \mathbb{E}_{q_{\theta_q}}[\log p(\mathbf{y} \mid \mathbf{u})]$ is the so-called *reconstructionterm*. From this it is clear that the derivative of $\mathcal{L}$ w.r.t. the variational parameters is

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_q} &= \frac{\mathrm{d}r}{\mathrm{d}\theta_q} - \eta\left(\theta_q\right) + \frac{\mathrm{d}\mu}{\mathrm{d}\theta_q}\left(\theta_0 - \theta_q\right) + \frac{\mathrm{d}A}{\mathrm{d}\theta_q} \\ &= \frac{\mathrm{d}r}{\mathrm{d}\theta_q} - \eta\left(\theta_q\right) + \mathbf{F}\left(\theta_q\right)\left(\theta_0 - \theta_q\right) + \eta\left(\theta_q\right) \\ &= \frac{\mathrm{d}r}{\mathrm{d}\theta_q} + \mathbf{F}\left(\theta_q\right)\left(\theta_0 - \theta_q\right). \end{aligned} \tag{2.38}$$

Substituting this result into the natural gradient step (equation 2.36) yields

$$\begin{aligned}\theta_q^{(t+1)} &= \theta_q^{(t)} + l^{(t+1)}\left[\mathbf{F}^{(t)}\right]^{-1}\left[\mathbf{F}^{(t)}\left(\theta_0 - \theta_q^{(t)}\right) + \left.\frac{\mathrm{d}r}{\mathrm{d}\theta_q}\right|_{\theta_q^{(t)}}\right] \\ &= \theta_q^{(t)} + l^{(t+1)}\left[\left[\mathbf{F}^{(t)}\right]^{-1}\mathbf{F}^{(t)}\left(\theta_0 - \theta_q^{(t)}\right) + \left[\mathbf{F}^{(t)}\right]^{-1}\left.\frac{\mathrm{d}r}{\mathrm{d}\theta_q}\right|_{\theta_q^{(t)}}\right] \\ &= \theta_q^{(t)} + l^{(t+1)}\left[\left(\theta_0 - \theta_q^{(t)}\right) + \left.\frac{\mathrm{d}\theta_q}{\mathrm{d}\eta}\right|_{\eta_t}\left.\frac{\mathrm{d}r}{\mathrm{d}\theta_q}\right|_{\theta_q^{(t)}}\right] \\ &= \left[1 - l^{(t+1)}\right]\theta_q^{(t)} + l^{(t+1)}\left[\theta_0 + \left.\frac{\mathrm{d}r}{\mathrm{d}\eta}\right|_{\eta_t}\right] \end{aligned} \tag{2.39}$$

where $\eta_t := \eta\left(\theta_q^{(t)}\right)$.

Chapter 3

Efficiently parameterised inducing point VI for GPs using stochastic natural gradients

3.1 Efficient $O(M)$ parameterisation of the approximate posterior

This project develops the prior works on sparse variational GP models (Hensman et al., 2013; Titsias, 2009), first, by introducing an alternative paramterisation to the approximate posterior. In the prior work, the approximate posterior is parameterised in terms of a mean vector $\mathbf{m}$ and a dense covariance matrix $\mathbf{S}$, earning $M + M(M+1)/2$ variational paramters (i.e. M parameters in $\mathbf{m}$ and $M(M+1)/2$ parameters in $\mathbf{S}$ since $\mathbf{S}$ has a unique Cholesky decomposition: $\mathbf{S} = \mathbf{L}\mathbf{L}^\top$).

$$\text{Original parameterisation:} \quad q(\mathbf{u}) := \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$$

As an alternative, we can parameterise the precision matrix of the approximate posterior $\mathbf{S}^{-1}$ as the sum of prior precision $\mathbf{K}_{mm}^{-1}$ and a positive-definite diagonal matrix $\mathbf{G}$ and the mean vector $\mathbf{m}$ as the multiplication of the prior covariance matrix $\mathbf{K}_{mm}$ and a mean vector in the affine transformation space μ. This parameterisation allows us to reduce the number of variational parameters from $M + M(M+1)/2$ to just $2M$ (i.e. M in μ and M in $\mathbf{diag}(\mathbf{G})$). In this paper we refer to this alternative parameterisation as the efficient $O(M)$ parameterisation

of the approximate posterior.

$$\text{Alternative parameterisation:} \quad q(\mathbf{u}) := \mathcal{N}\left(\mathbf{u}; \mathbf{K}_{mm}\mu, \left[\mathbf{K}_{mm}^{-1} + \mathbf{G}\right]^{-1}\right)$$

Futhermore, we combine this variational method with the stochastic natural gradient descent algorithm (Amari, 1998). In this paper, we refer to our sparse variational GP model with efficiently parameterised approximate posterior combined with NGD as our (NGD) method.

In return for reducing the number of variational parameters from $M + M(M+1)/2$ to just $2M$, the likelihood of obtaining the optimal approximate posterior is expected to decline. We consider that the loss in accuracy may be small, and that we should benefit from improved numerical stability during training and faster convergence.

As far as we know, the only other work on this efficient $O(M)$ parameterisation was conducted by Panos et al. (2018). In this prior work, the experiments presented did not use natural gradients, and leave open questions around the overall performance of such parameterisation.

3.1.1 Motivation

The motivation behind our efficient $O(M)$ parameterisation is basically the computational cost. Panos et al. (2018) mentioned that all previous work on sparse variational GPs, including the two methods introduced in the previous section (Hensman et al., 2013; Titsias, 2009), do not take into account the dimensionality of the input space D when expressing time complexities and somehow D is assumed to be small or of the order of M. Since D appears in the lower bound only through the computation of the covariance matrix $\mathbf{K}_{mm}$ and the cross covariance matrix $\mathbf{K}_{n'm}$, where n' is the size of minibatch which scales as $O(M)$, the time complexity with respect to D is clearly $O\left(DM^2\right)$ since evaluating any standard kernel function on each pair of instances scales as $O(D)$. Thus, each optimization step of the lower bound actually scales overall as $O\left(DM^2 + M^3\right)$ and when D is larger than M the term $O\left(DM^2\right)$ dominates. For instance, in a dataset as MNIST where $D = 784$ and $M = 500$ the optimization of the bound will roughly be of order $O\left(M^3\right)$, while in other datasets with even slightly larger D, such as the CIFAR-10 dataset where $D = 3072$, the term $O\left(DM^2\right)$ dominates and thus results in slower training. Thus, we need more computationally economical parameterisation for the approximate posterior $q(\mathbf{u})$.

According to Opper and Archambeau (2009), when not utilising pseudo-points (i.e. non-sparse variational GP models), the optimal GP variational approximation to the posterior

is given as follows:

$$q(f) = q(\mathbf{f})p(f_{\neq\mathbf{f}}|\mathbf{f}) \quad q(\mathbf{f}) := \mathcal{N}\left(\mathbf{f}; \mathbf{m}, \left[\mathbf{K}_{nn}^{-1} + \hat{\mathbf{G}}\right]^{-1}\right) \tag{3.1}$$

where $\mathbf{m}$ is a vector, $\mathbf{K}_{nn}^{-1}$ is a prior precision matrix, and $\mathbf{G}$ is a positive-definite diagonal matrix. This approximate posterior is equivalent to the exact posterior obtained under a Gaussian likelihood whose observations and variance are a function of $\mathbf{m}$ and $\mathbf{G}$. This approximation requires $2N$ variational parameters causing the GP scaling problem. To address this issue, we combine this parameterisation with a pseudo-point approximation (Hensman et al., 2013). Let us partition f into two different groups of variables: an M dimensional vector $\mathbf{u}$ (relevant to inducing inputs) where $M < N$, and the others in $f_{\neq\mathbf{u}}$.

$$q(f) = q(\mathbf{u})p\left(f_{\neq\mathbf{u}} \mid \mathbf{u}\right), \quad q(\mathbf{u}) := \mathcal{N}\left(\mathbf{u}; \mathbf{K}_{mm}\mathbf{m}, \left[\mathbf{K}_{mm}^{-1} + \mathbf{G}\right]^{-1}\right) \tag{3.2}$$

where $\mathbf{m}$ is replaced by $\mathbf{K}_{mm}\mathbf{m}$ so that it will be in the same affine transformation space as the vec$\,\mathbf{G}$ (see section 3.2 for detailed explanation). We consider that we can still benefit from this computationally economical parameterisation with the use of pseudo-point approximations.

We now present the motivation behind the use of the natural gradient combined with the alternative parameterisation. NGD is invariant to the parameterisation of the target distribution. Further, its applicability to a sparse variational GP model in the conjugate case is shown by Hensman et al. (2013) and is explored in the non-conjugate stochastic settings by Salimbeni et al. (2018). Thus, we consider that, with our alternative parameterisation, we can still benefit from the stability and fast convergence of NGD, and possibly we can further accelerate, stabilise, and make more robust NGD. However, in practice, we cannot use the natural gradient directly for the hyperparameter optimization because we do not have a probability distribution for the hyperparameters. Following the suggestion by Salimbeni et al. (2018), instead of direct use of natural gradients, we use a sequential scheme where we perform a step of Adam on the hyperparameters, followed by a step of NGD on the variational parameters. See section 4.1.1 for a detailed explanation of the implementation.

3.1.2 Theoretical understanding

In variational inference, the most commonly used parameterisation is in terms of the mean and covariance $\mathbf{m}, \mathbf{S}$, where $\mathbf{S}$ is further parameterised based on the Cholesky decomposition $\mathbf{S} = \mathbf{L}\mathbf{L}^\top$ to maintain positivity of the covariance. However, this parameterisation can lead to slow convergence due to the strong dependence of $\mathbf{m}, \mathbf{S}$ with the kernel matrix $\mathbf{K}_{mm}$ from the prior $p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{mm})$. To highlight this dependence, we first rewrite the lower bound

$\mathcal{L}_2$ in Hensman et al. (2013) as follows (see equation 2.24 and 2.20 for the definition of $\mathcal{L}_2$ and $\mathcal{L}_1$ respectively):

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{X}) &\geq \mathcal{L}_2 \\
&= \mathbb{E}_{q(\mathbf{u})}\left(\mathcal{L}_1 + \log p(\mathbf{u}) - \log q(\mathbf{u})\right) \\
&= \int \left(\mathcal{L}_1 + \log p(\mathbf{u}) - \log q(\mathbf{u})\right) q(\mathbf{u}) d\mathbf{u} \\
&= \int \mathcal{L}_1 q(\mathbf{u}) d\mathbf{u} - \int q(\mathbf{u}) \log \frac{q(\mathbf{u})}{p(\mathbf{u})} d\mathbf{u} \\
&= \int \int \left(\log p(\mathbf{y} \mid \mathbf{f})\right) p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) d\mathbf{f} d\mathbf{u} - \mathrm{KL}(q(\mathbf{u}) || p(\mathbf{u})) \\
&= \mathbb{E}_{q(\mathbf{f})}\left[\log p(\mathbf{y} \mid \mathbf{f})\right] - \mathrm{KL}(q(\mathbf{u}) || p(\mathbf{u})) \\
&= \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{u})}\left[\log p(y_i \mid f_i)\right] - \mathrm{KL}(q(\mathbf{u}) || p(\mathbf{u}))
\end{aligned} \tag{3.3}$$

where $q(f_i) = \int p(f_i \mid \mathbf{u}) q(\mathbf{u}) d\mathbf{u}$.

We can further rewrite this lower bound by marginalizing out $\mathbf{f}$ instead of marginalizing out $\mathbf{u}$ as follows:

$$\log p(\mathbf{y} \mid \mathbf{X}) \geq \sum_{i=1}^{N} \mathbb{E}_{q(f_i)}\left[\log \mathcal{F}(y_i, \mathbf{u})\right] - \mathrm{KL}(q(\mathbf{u}) || p(\mathbf{u})) \tag{3.4}$$

where $\log \mathcal{F}(y_i, \mathbf{u}) = \mathbb{E}_{q(f_i|\mathbf{u})}\left[\log p\left(y_i, \mathbf{u}\right)\right]$.

A straightforward derivation similar to the proof in Opper and Archambeau (2009) can reveal that at maximum it holds $\mathbf{m} = \mathbf{K}_{mm}\mu$ and $\mathbf{S} = \left(\mathbf{K}_{mm}^{-1} + \Lambda^{-1}\right)^{-1}$ for some vector μ and some full (non-diagonal) positive definite matrix Λ associated with the second derivatives of the first data term in the above bound. Given this, we parameterise $q(\mathbf{u})$ in terms of (μ, Λ) so that we only need to compute the prior covariance matrix $\mathbf{K}_{mm}$ once at the beginning of the process. However, this can still lead to a slow optimization because there are $O\left(M^2\right)$ variational parameters to be optimized in the full Λ matrix. Therefore, we propose to simplify this parameterisation by replacing Λ with a diagonal covariance matrix $\mathbf{G}$ leading to our efficient $O(M)$ parameterisation shown in

$$\mathbf{m} = \mathbf{K}_{mm}\mu, \quad \mathbf{S} = \left(\mathbf{K}_{mm}^{-1} + \mathbf{G}\right)^{-1} = \mathbf{K}_{mm} - \mathbf{K}_{mm}\left(\mathbf{K}_{mm} + \mathbf{G}^{-1}\right)^{-1}\mathbf{K}_{mm} \tag{3.5}$$

where $\mu \in \mathbb{R}^M$ is a real-valued vector of variational parameters and $\mathbf{G}$ is a positive diagonal matrix (i.e. each diagonal element is constrained to be positive) parameterised by M variational parameters. Thus, overall $q(\mathbf{u})$ is parameterised by $2M$ variational parameters while

all the remaining structure comes from a careful preconditioning with the prior covariance matrix $\mathbf{K}_{mm}$.

Note that we can recover the expression of $q(\mathbf{f}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{m}_f, \mathbf{S}_f\right)$ using the efficient parameterisation of $q(\mathbf{u})$ as:

$$\mathbf{m}_f = \mathbf{K}_{nn}\mu, \quad \mathbf{S}_f = \mathbf{K}_{nn} - \mathbf{K}_{nm}\left(\mathbf{K}_{mm} + \mathbf{G}^{-1}\right)^{-1}\mathbf{K}_{mn} \tag{3.6}$$

which can recover the optimal $q^*(\mathbf{f})$ when $M = N$ inducing inputs are placed at the location of the training inputs (i.e. $\mathbf{Z} = \mathbf{X}$). In other cases, the restricted covariance in $q(\mathbf{f})$ will not be able to match exactly the optimal $q^*(\mathbf{f})$, but in practice it tends to be very flexible especially when we optimize over the inducing inputs Z so that a posteriori $\mathbf{f}$ is well reconstructed by $\mathbf{u}$.

Furthermore, the above parameterisation of $q(\mathbf{u})$ leads to a numerically stable and simplified form of the lower bound. Specifically, the KL divergence term in 3.3 reduces to

$$\mathrm{KL}[q(\mathbf{u})\|p(\mathbf{u})] = \frac{1}{2}\mu^\top \mathbf{K}_{mm}\mu - \frac{1}{2}\mathrm{tr}\left(\left(\mathbf{K}_{mm} + \mathbf{G}^{-1}\right)^{-1}\mathbf{K}_{mm}\right) + \frac{1}{2}\log\left|\mathbf{K}_{mm} + \mathbf{G}^{-1}\right| - \frac{1}{2}\log|\mathbf{G}^{-1}| \tag{3.7}$$

while each marginal $q(f_i)$ in the expectations of the first data term in 3.3 becomes $q(f_i) = \mathcal{N}(f_i \mid m_i, s_i)$ where m_i and s_i are the i -th elements of the vectors $\mathbf{m}_f$ and $\mathbf{S}_f$ in 3.6. Therefore, the overall bound in 3.3 obtains a simplified and numerically stable form because of the cancellation of all inverses and determinants of $\mathbf{K}_{mm}$. At each optimization the only matrix we need to decompose using Cholesky decomposition is $\mathbf{K}_{mm} + \mathbf{G}^{-1}$, which is in an already numerically stable form due to the inflation of the diagonal of $\mathbf{K}_{mm}$ with $\mathbf{G}^{-1}$. In practice we add a small jitter to the variational parameters $\mathrm{diag}(\mathbf{G})$ to ensure numerical stability throughout optimization (in our experiments the jitter level is set to 10^{-10}).

3.2 Affine transformation and the natural parameters

This section explains how we perform NGD on our efficient $O(M)$ parameterisation. To perform NGD, the natural parameters are required. Thus, in the original parameterisation $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$, we compute the natural parameters from the mean and covariance matrix as explained in the section 2.2.3 and 2.3.1. Instead, in our alternative parameterisation $q(\mathbf{u}) := \mathcal{N}\left(\mathbf{u}; \mathbf{K}_{mm}\mu, \left[\mathbf{K}_{mm}^{-1} + \mathbf{G}\right]^{-1}\right)$, NGD is performed directly on the new variational parameters μ and $\mathrm{diag}(G)$ by constructing a new exponential family for which μ and $\mathrm{diag}(G)$ are the natural parameters.

First, let us consider a general case of parameterising the natural parameter θ in terms of some other parameter β. Specifically, consider

$$\theta(\beta) := C\beta + c \tag{3.8}$$

for some linear transformation C (e.g. a matrix) and vector c. The corresponding exponential family density can be expressed in terms of β :

$$\begin{aligned} p(\mathbf{u}) &= h(\mathbf{u}) \exp(\langle \theta(\beta), \phi(\mathbf{u}) \rangle - A(\theta(\beta))) \\ &= h(\mathbf{u}) \exp(\langle C\beta + c, \phi(\mathbf{u}) \rangle - A(C\beta + c)) \\ &= h(\mathbf{u}) \exp\left(\langle c, \phi(\mathbf{u}) \rangle + \langle \beta, C^* \phi(\mathbf{u}) \rangle - A(C\beta + c)\right) \end{aligned} \tag{3.9}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, $h : \mathbb{R}^P \to \mathbb{R}$ is the base measure, $\phi : \mathbb{R}^P \to \mathbb{R}^D$ is the sufficient statistic function, $A : \mathbb{R}^D \to \mathbb{R}$ is the log partition function, and C^* is the adjoint of C, which is equivalent to $C^\top$ if C is a real-valued matrix. Now let

$$h_\beta(\mathbf{u}) := h(\mathbf{u}) \exp(\langle c, \phi(\mathbf{u}) \rangle) \tag{3.10}$$

$$\phi_\beta(\mathbf{u}) := C^\top \phi(\mathbf{u}) \tag{3.11}$$

$$A_\beta(\beta) := A(C\beta + c) \tag{3.12}$$

Expressing equation 3.9 in terms of these quantities yields another exponential family density:

$$p_\beta(\mathbf{u}) := h_\beta(\mathbf{u}) \exp\left(\left\langle \beta, \phi_\beta(\mathbf{u}) \right\rangle - A_\beta(\beta)\right) \tag{3.13}$$

where β is the natural parameter.

Now, let us consider a specific case of our parameterisation. The natural parameter of the approximate posterior can be parameterised as the precision matrix $\mathbf{S}^{-1}$ as follows:

$$\theta = [\theta_1, \theta_2] = \left[\mathbf{S}^{-1}\mathbf{m}, \operatorname{vec}\left(-\frac{1}{2}\mathbf{S}^{-1}\right)\right] \tag{3.14}$$

Let $c \in \mathbb{R}^{D^2}$ and $C \in \mathbb{R}^{D^2 \times D}$ be the matrix such that

$$c := \operatorname{vec}(\mathbf{K}_{mm}^{-1}) \tag{3.15}$$

$$\operatorname{vec}(\mathbf{G}) = C \operatorname{diag}(\mathbf{G}) \tag{3.16}$$

Then, using the definition of $\mathbf{G}$, the precision component of the natural parameter in equation 3.14 can be written as:

$$\text{vec}\left(\mathbf{S}^{-1}\right) = c + C\mathbf{g} \tag{3.17}$$

Thus, our new parameter $\text{diag}(\mathbf{G})$ is the natural parameter in an affine subspace and can be used for NGD performed in this affine subspace. Similarly, we can use our new parameter μ as the natural parameter in this affine subspace.

To perform NGD efficiently, we need the expectation parameter in the affine transformation space as discussed in section 2.3. Using the definition for the expectation parameters in equation 2.32, it can be derived using the expectation parameter in the original space:

$$\begin{aligned}
{\eta_{affine}}^{\top} &= \frac{\partial A_{affine}}{\partial \theta_{affine}} \\
&= \frac{\partial A}{\partial \theta}\frac{\partial \theta}{\partial \theta_{affine}} \\
&= \eta^{\top}\frac{\partial \theta}{\partial \theta_{affine}}
\end{aligned} \tag{3.18}$$

This means that the relationship of the expectation parameters η, η_{affine} can be represented by the mapping between the natural parameters in the affine transformation space and the original space. Thus, the gradient with respect to the expectation parameters is given as follows:

$$\frac{\partial \eta}{\partial \eta_{affine}} = \left(\frac{\partial \theta_{affine}}{\partial \theta}\right)^{\top} \tag{3.19}$$

Thus, using the above equation, the natural gradient can be calculated as follows:

$$\begin{aligned}
\tilde{\nabla}_{\boldsymbol{\theta}_{affine}}\mathcal{L} &= \frac{\partial \mathcal{L}}{\partial \eta_{affine}} \\
&= \frac{\partial \mathcal{L}}{\partial \eta}\frac{\partial \eta}{\partial \eta_{affine}} \\
&= \frac{\partial \mathcal{L}}{\partial \eta}\left(\frac{\partial \theta_{affine}}{\partial \theta}\right)^{\top}
\end{aligned} \tag{3.20}$$

Chapter 4

Experiments and discussion

In this chapter we investigate the efficiency of the natural gradient method in a sparse variational GP model, and explore the practical applicability of our efficient $O(M)$ parameterisation for the approximate variational posterior. We start by explaining the implementation of the methods and the crucial step of parameter initialisation. We then evaluate the performance of the models, firstly on several one-dimensional synthetic datasets, and secondly on a real dataset.

4.1 Experimental methods

4.1.1 Natural Gradient Descent

The attractive features of natural gradients are well understood. However, as far as we know, the conditions where natural gradients outperform ordinal gradients was, to date, unexplored. Thus, this project seeks to substantiate the advantages of natural gradients under a number of settings as well as confirming the superiority of natural gradients.

However, in practice, there is an issue in hyperparameter optimization using natural gradients. As we do not have a probability distribution for the hyperparameters, we cannot use natural gradients directly for the hyperparameter optimization. Thus, following the suggestion by Salimbeni et al. (2018), we perform a full stochastic optimization using natural gradients in the following way: a first step using Adam to optimize the hyperparameters (with step size γ_{adam}, followed by a second step using NGD to optimize the variational parameters $\mathbf{m}, \mathbf{S}$ (with step size γ). In this project, this sequential two step scheme is referred as NGD optimization. For a comparison, we will use Adam to simultaneously optimize both variational parameters and hyperparameters in a single step, referred to as Adam optimization.

4.1.2 Efficient $O(M)$ parameterisation of the approximate posterior

Following the investigation on the natural gradients, we test our method, a pseudo-point approximation with the efficient $O(M)$ parameterisation of the approximate posterior combined with NGD. As discussed in the previous chapter, we parameterise the Gaussian variational posterior using our alternative variational parameters μ and $\mathrm{diag}(\mathbf{G})$, and perform NGD on them considering they are the natural parameters in an affine transformation space.

Again, a stochastic variational inference is performed sequentially: a first step using Adam to optimize the hyperparameters, followed by a second step using NGD to optimize the variational parameters which is now μ and $operatornamediag(\mathbf{G})$ instead of $\mathbf{m}$ and $\mathbf{S}$.

4.1.3 Datasets and parameter initialisation

For the experiments, we need a training dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{N \times D}$ is the set of training inputs, and $\mathbf{y} = \{y_i\}_{i=1}^N \in \mathbb{R}^N$ is the corresponding target vector. This section presents three different datasets which we will repeatedly use throughout the experiments (a synthetic sinusoidal, a synthetic GP, and the NAVAL dataset). Furthermore, we show how parameters are initialised. Regardless of the datasets, we use a small jitter level of 10^{-10}, and a Gaussian likelihood where the likelihood variance is initialised to 0.1 based on our empirical research. The step size of Adam is set to 10^{-2} if not otherwise notated.

Synthetic sinusoidal dataset

The function below is used to produce the synthetic sinusoidal dataset:

$$f(x) = \sin(3x\pi) + 0.3\cos(9x\pi) + 0.5\sin(7x\pi)$$

The training inputs $N = 100$ are drawn from a uniform distribution on $[-1, 1]$. The observation is drawn from the function above and the observation noise with variance $\beta^{-1} = 10^{-2}$ is added. The inducing inputs $M = 10$ are initialised with k-mean. We use an SE kernel (see equation 2.3) with kernel length scale and kernel variance initialised based on the scheme in Ulapane et al. (2020).

Synthetic GP dataset

As another simple toy dataset we use a synthetic GP dataset. The training inputs $N = 100$ are drawn from a uniform distribution on $[-1, 1]$. The observation is drawn jointly from a single GP whose kernel is an SE kernel with kernel length scale $\ell = 0.1$ and kernel variance $\sigma_{SE} = 0.5$. The observation noise with variance $\beta^{-1} = 10^{-2}$ is added. The inducing inputs

$M = 10$ are initialised with k-mean. We use an SE kernel with kernel length scale and kernel variance initialised based on the scheme in Ulapane et al. (2020).

Real dataset: NAVAL

To show the stability of natural gradients in ill-conditioned settings, we employ the NAVAL dataset (Crisher and Souva, 2013) as our real dataset. In ill-conditioned settings ordinary gradients suffer from instability (Sun et al., 2009) and slow convergence. As the natural gradient is invariant to parameterisation, NGD should not be adversely affected by issues of conditioning. The NAVAL dataset contains $N = 11000$ training inputs with $D = 16$ dimensions, and target values uniformly distributed in 51 increments between 0.95 and 1. Following the settings in Salimbeni et al. (2018), we use a minibatch size of 256, the inducing inputs of $M = 100$, and a Matern $5/2$ kernel (see equation 2.4) with the length scale initialised to the square root of the data dimension $\ell = \sqrt{D} = 4$ and the kernel variances initialised to $\sigma_{Matern} = 2$. We use a Gaussian likelihood with the likelihood variance initialised to $\sigma_{likelihood} = 0.1$. The step size of NGD is scheduled to begin with $\gamma_{\text{initial}} = 10^{-4}$ and be increased through 40 iterations to $\gamma_{\text{final}} = 10^{-1}$. For the NAVAL dataset, we apply data normalization explained in detail below.

Data normalisation

A common data normalisation scheme is to scale the training dataset $\mathcal{D} = (\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{y} \in \mathbb{R}^{N})$ to have zero mean and unit covariance. Suppose $\mu_x = \left[\mu_1, \mu_2, \ldots, \mu_j, \ldots, \mu_D\right]$ is a row vector where $\mu_j, j = 1, 2, \ldots, D$ is the mean of the j^{th} column of $\mathbf{X}$, and $\sigma_x = \left[\sigma_1, \sigma_2, \ldots, \sigma_j, \ldots, \sigma_D\right]$ is a row vector where $\sigma_j, j = 1, 2, \ldots, N$ is the standard deviation of the j^{th} column of $\mathbf{X}$. Now normalization is done for all j as follows.

$$(X)_{*,j} \leftarrow \frac{(X)_{*,j} - \mu_j}{\sigma_j} \tag{4.1}$$

Similarly, suppose μ_y and σ_y are the mean and standard deviation respectively, of the target vector $\mathbf{y} = \{y_i\}_{i=1}^{D}$. Now, normalization of $\mathbf{y}$ is done by performing

$$y_i \leftarrow \frac{y_i - \mu_y}{\sigma_y}$$

Kernel initialisation scheme

Initialisation of kernels is important because, if variance of kernels over outputs is much smaller than that of kernels over inputs, the model could have difficulties in finding the expected correlations between outputs. This project uses the initialisation scheme introduced in Ulapane et al. (2020) for SE kernels.

The set of hyperparameter initial values for the kernel length scale and kernel variance is denoted as $\{\sigma_{SE}, \ell_{SE}\}$.

To start with, constructing sets $S_1, S_2, \ldots, S_D$ is proposed where $S_j = \{\mathbf{y}, \mathbf{X}_{*,j}\}$ for $j = 1, 2, \ldots, D$. $\mathbf{X}_{*,j}$ is the j^{th} column of $\mathbf{X}$. All $S_j, j = 1, 2, \ldots, D$ are to be rearranged such that $\mathbf{X}_{*,j}$ will be sorted to be in ascending order, and $\mathbf{y}$ will be sorted correspondingly. From this point onward, $\widetilde{\mathbf{X}}_{*,j}$ would represent the sorted j^{th} column of $\mathbf{X}$, and $\widetilde{\mathbf{y}}_j$ would represent correspondingly sorted $\mathbf{y}$, and S_j would represent a sorted set. Next, constructing the set $d_x = \left[d_{x1}, d_{x2}, \ldots, d_{xj}, \ldots, d_{xD}\right]$ is proposed where d_{xj} for $j = 1, 2, \ldots, D$ is given by

$$d_{xj} = \frac{1}{k_{xj}} \sum_{i=1}^{N-1} \left| \widetilde{\mathbf{X}}_{i,j} - \widetilde{\mathbf{X}}_{i+1,j} \right|$$

where k_{xj} is the number of instances that $\widetilde{\mathbf{X}}_{i,j} \neq \widetilde{\mathbf{X}}_{i+1,j}$ for $i = 1, 2, \ldots, N-1$. Then, ℓ_{SE} is set to be the mean of d_x and naturally $\ell_{SE} > 0$ condition would hold.

To determine σ_{SE}, constructing the set $d_y = \left[d_{y1}, d_{y2}, \ldots, d_{yj}, \ldots, d_{yD}\right]$ is proposed where d_{yj} for $j = 1, 2, \ldots, D$ is given by

$$d_{yj} = \frac{1}{k_{yj}} \sum_{i=1}^{m-1} \left| \left(\widetilde{\mathbf{y}}_j\right)_i - \left(\widetilde{\mathbf{y}}_j\right)_{i+1} \right|$$

where $\left(\widetilde{\mathbf{y}}_j\right)_i$ is the i^{th} element of $\widetilde{\mathbf{y}}_j$ and k_{yj} is the number of instances where $\left(\widetilde{\mathbf{y}}_j\right)_i \neq \left(\widetilde{\mathbf{y}}_j\right)_{i+1}$ for $i = 1, 2, \ldots, N-1$. Then σ_{SE} is set to be the mean of d_y. There lies a possibility of $\sigma_{SE} = 0$ occurring, and if σ_{SE} becomes zero, the covariance function values would become zero. Therefore, $\sigma_{SE} \neq 0$ should hold for the covariance function to produce meaningful values. This imposes that an exception handling routine should come into effect in the rare event of $\sigma_{SE} = 0$ occurring, to indicate that the available training data set is unsuitable to proceed with GP regression, and that more training points are required to make sure $\sigma_{SE} \neq 0$ results.

4.2 Experimental results

4.2.1 Overview

All experiments described in this chapter utilize the library GPflow (Matthews et al., 2017), which takes advantage of the benefits of TensorFlow 2.0. The performance is evaluated in terms of the progress in the ELBO and hyperparameter values, as well as the output prediction samples in the cases of synthetic datasets.

The hyperparameters contain the kernel variance s_f^2, the kernel length scale ℓ and the noise variance σ_n^2. For hyperparameter optimization, we use the Adam (Kingma and Ba, 2014) optimizer with default parameter values and a fixed learning rate of 0.01, if not otherwise notated. In the experiments on NGD in sparse variational models in section 4.2.2, the variational parameters contain a mean vector $\mathbf{m}$ and a covariance matrix which is further parameterised by its Cholesky decomposition $\mathbf{S} = \mathbf{L}\mathbf{L}^{top}$. They are initialised as a zero mean vector and a unit covariance matrix. Whereas, in the experiments on our alternative parameterisation in section 4.2.3, the variational parameters contain μ and $\mathrm{diag}(\mathbf{G})$. They are initialised as a zero mean vector and a vector filled with ones so that $\mathbf{G}$ will be a unit matrix.

We use the Gaussian likelihoods for all experiments, with the likelihood variance initialised to 0.1 if not otherwise notated.

4.2.2 Natural Gradient Descent

This section investigates the performance of a sparse variational GP model with natural gradient descent, compared with a sparse variational GP with a standard gradient descent algorithm, Adam.

On a small synthetic sinusoidal data set, we first compare the quality of direction and the speed of convergence. The models try to optimize the variational parameters $\mathbf{m}, \mathbf{S}$ while the hyperparameters are fixed to the optimum values. Secondly, we free the hyperparameter constraint so that the models conduct full variational inference optimizing both variational parameters and hyperparameters simultaneously. We compare the performances of NGD and Adam while ensuring the applicability of NGD to a full variational inference. Thirdly, we investigate under which conditions natural gradients exhibit superiority in performing the optimization in different settings (e.g. number of inducing inputs, number of training points, minibatch size, and noise variance). Finally, using a real dataset (the NAVAL dataset Crisher and Souva (2013)) we will confirm the applicability of the models and compare their stability in ill-conditioned settings.

Synthetic Data Experiments: stochastic optimization with fixed hyperparameters

Firstly, we concentrate on optimization of the variational parameters whilst setting the hyperparameters to optimum values based on our empirical research. Each hyperparameter is fixed as follows: the kernel length scale $\ell_{SE} = 0.813$, kernel variance $\sigma_{SE} = 0.112$, and noise variance $\sigma_{noise} = 0.045$.

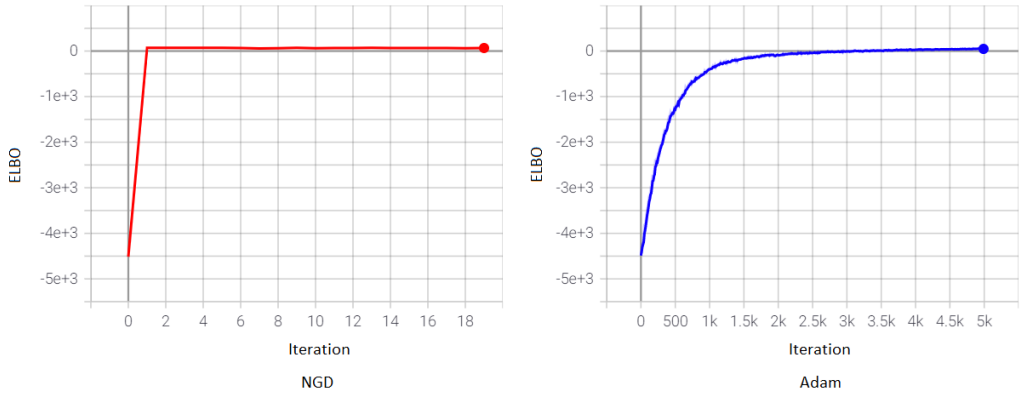


Fig. 4.1 The stochastic optimization of the lower bound for fixed hyperparameters in the case of $N = 300, M = 10$, no use of minibatch, noise variance 0.01, and hyperparameters set to the optimum values. NGD (shown in red) takes only a single iteration to reach the optimum when the step size is 1, whereas Adam (shown in blue) takes thousands of iterations.

Figure 4.1 shows the results of the sparse variational GP models, one using the NGD algorithm and another using the Adam optimizer. Agreeing with the prior work (Hensman et al., 2013), in the case of the Gaussian likelihood, NGD finds the optimum in a single step when the step size $\gamma = 1$, whereas it takes thousands of iterations for the Adam optimizer. This result confirms the superiority of natural gradients in the stochastic setting of a sparse variational GP model. This result also implies that natural gradients provide a superior quality of direction.

Synthetic Data Experiments: full stochastic optimization

Secondly, letting the hyperparameters be optimized by Adam, we perform full stochastic optimization using NGD. As discussed before, we compare the use of Adam to simultaneously optimize both variational parameters and hyperparameters in a single step, with a sequential two step optimization, first on hyperparameters using Adam and then on variational parameters using natural gradients.

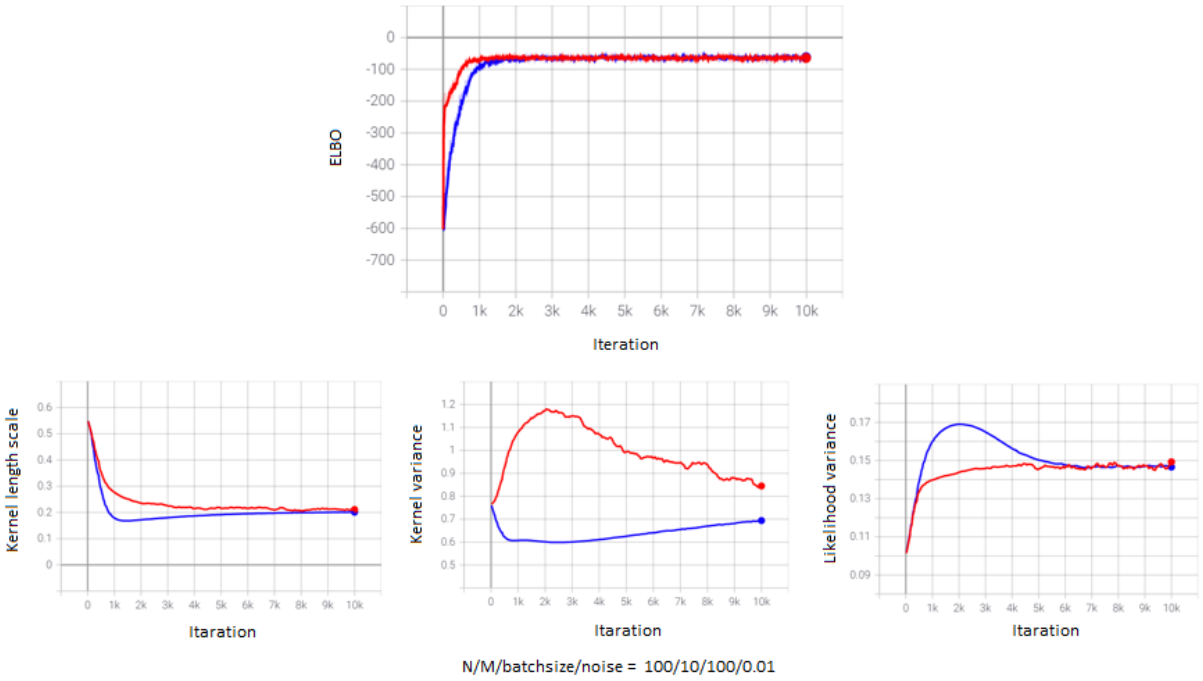


Fig. 4.2 Joint optimization of the hyperparameters and the variational distribution in the case of $N = 100, M = 10$, no use of minibatch, and noise variance 0.01. NGD (shown in red) and Adam (shown in blue) take different paths for the optimization process, reaching the same level of the ELBO.

Figure 4.2 shows the results of full stochastic optimization of the variational distribution with NGD or Adam. Once again, NGD outperforms the Adam optimizer in terms of speed of convergence, reaching the optimum in under 1000 iterations (see the ELBO graph). The graphs illustrating the optimization process of the hyperparameters show that even after the ELBOs have converged, the values of the hyperparameters change a lot, searching for a better optimum. This means that the models have stabilised despite ongoing processes for hyperparameter optimization. We also note that, whilest not affecting the end result of the ELBO, the two models follow different path for the hyperparameters.

Synthetic Data Experiments: NGD v.s. Adam in different settings

Now we aim to provide evidence to answer "under what conditions does NGD converge faster than Adam". We perform expensive grid search on:

- The number of training inputs $N = 10/100/1000/10000$,
- number of inducing inputs $M = 10/100/1000$ where $M < N$,
- minibatch size $batchsize = 10/100/1000/10000$ where $batchsize <= N$,
- noise variance $noise = 0.1/0.01/0.001$.

We present the key results with a number of important findings listed below:

- The key factors are M and the batch size
- If M is small enough, NGD is always faster
- If M is large and we do not use a minibatch, NGD is faster in the beginning, but both converge after the same number of iterations
- Only, if M is large and minibatch is used, NGD is slower, but it still reaches the same optimum as Adam

Firstly, we demonstrate the case when M is small. In the synthetic sinusoidal dataset, let us consider $M = 10$ as appropriate based on our initial empirical experiments.

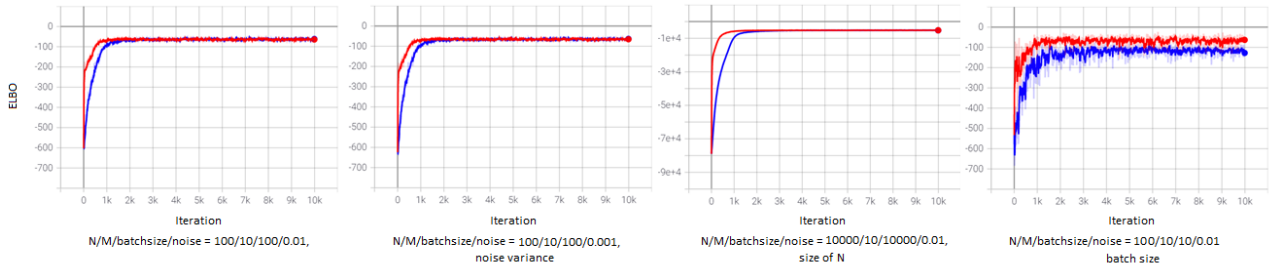


Fig. 4.3 Joint optimization of the hyperparameters and the variational distribution in different settings with M fixed to a small value $M = 10$. NGD (shown in red) is always faster than Adam (shown in blue) even when achieving a better result. Comparing the first two graphs, we see that the noise variance does not affect the relative performance of NGD and Adam. Similarly, comparing the first and the third, and the first and fourth, we see that neither the size of N nor the batch size affects the relative performance.

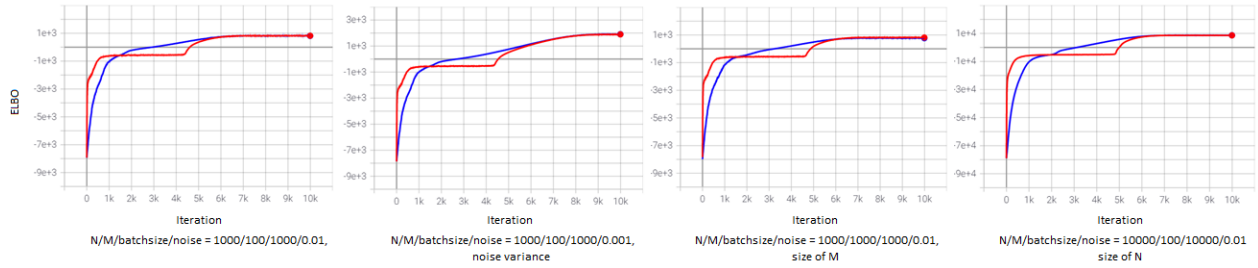


Fig. 4.4 Joint optimization of the hyperparameters and the variational distribution in different settings with M fixed to a large value $M \geq 100$ and without minibatch (i.e. $batchsize = N$). Both NGD and Adam converge after the same number of iterations, reaching the same level of ELBO. Comparing the first two figures, we see that the noise variance does not affect the relative performance of NGD and Adam. Similarly, comparing the first and the third, and the first and fourth, demonstrates that neither further increasing M nor increasing N affects the relative performance.

Figure 4.3 compares the results of ELBO in different settings with M equals to a small value 10. We can see that, regardless of N, batch size, or noise variance, NGD is always faster than Adam, reaching a better or identical level of ELBO.

Figure 4.4 compares the results of ELBO in different settings with M set to a relatively large value $M \geq 100$ and the batch size set to the size of the training data N (i.e. no use of minibatch). For a relatively large M, it appears that NGD and Adam not only end up at the same optimum at the same time, but also they follow a similar path in the latter stages of the optimization process. In all figures, NGD is a faster to begin with, then Adam is faster, then in the end, they follow the same path to the same level of ELBO.

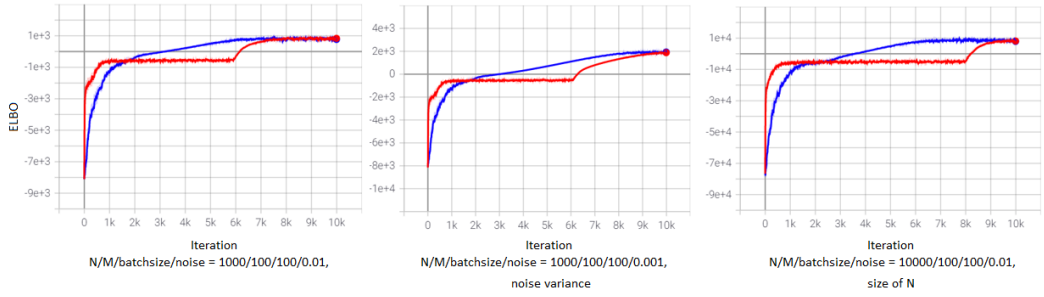


Fig. 4.5 Joint optimization of the hyperparameters and the variational distribution in different settings with M fixed to a large value $M = 100$ together with the use of minibatch. NGD (shown in red) is slower to converge, yet reaches the same level of ELBO as Adam (shown in blue). Comparing the first two figures, and the first and the third, we see that neither the noise variance nor the size of N affects the relative performance of NGD and Adam.

Figure 4.5 compares the results of ELBO in different settings with M set to a relatively large value while using a minibatch. Unfortunately, in this case, NGD's speed of convergence is slower than that of Adam, but NGD still finds the same optimum value for the ELBO. Note that it is not the use of minibatch per se, but the combination of the use of minibatch with a large value of M that results in NGD slower convergence. We can verify this by comparing the first and fourth figures in Figure 4.3.

To summarise the results of the grid search, NGD is essentially at least as good as Adam in terms of both the quality of optimization and the speed of convergence in typical practical settings where M is small and where we can benefit from the sparsity of the model.

Real-data: NAVAL Data Experiments

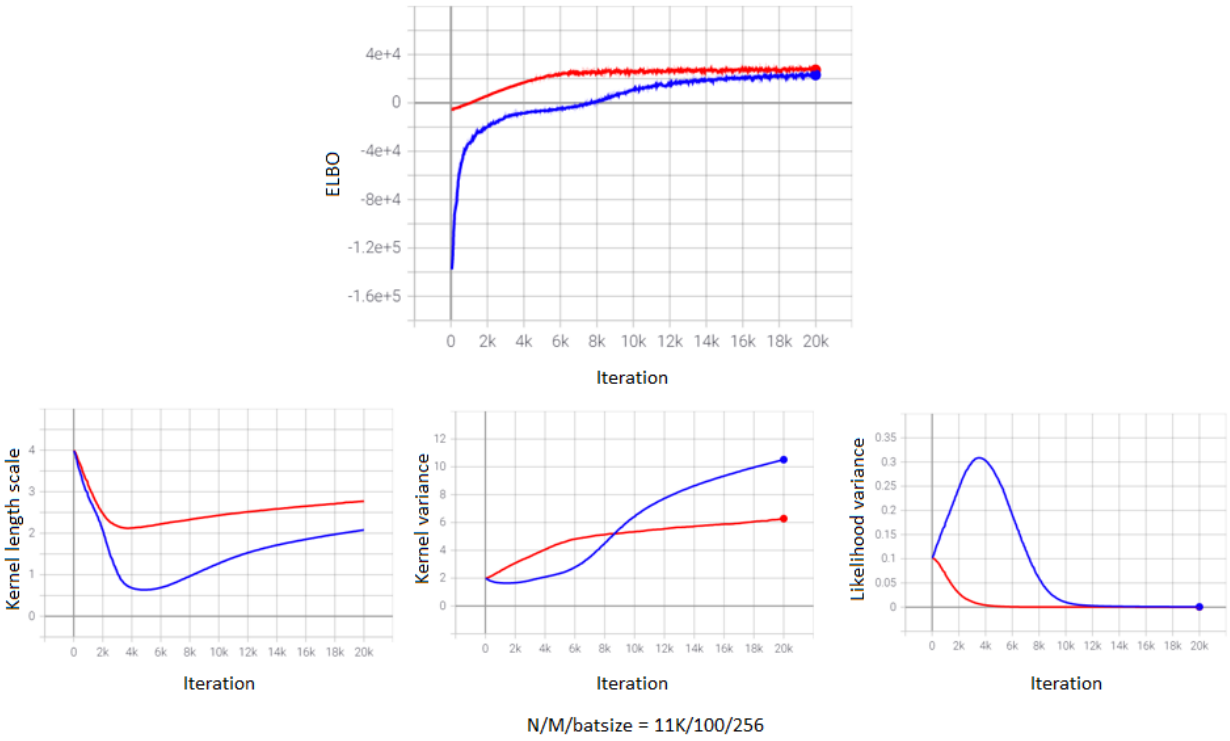


Fig. 4.6 Joint optimization of the hyperparameters and the variational distribution in the NAVAL dataset. NGD (shown in red) is faster to converge, whereas Adam (shown in blue) has not converged after a quite large number of iterations 20k.

Finally, using the NAVAL dataset we will confirm the applicability of the models to a large real dataset and compare the stability of the models in ill-conditioned settings. Following the setting in Salimbeni et al. (2018), we use $N = 11\text{k}$, $D = 16, M = 100$, minibatch size $= 256$, and run the optimization process for 20k iterations.

Figure 4.6 shows the results of stochastic optimization of the ELBO with respect to both the variational parameters and hyperparameters. Even given a large number of iterations, Adam cannot achieve the optimum whereas natural gradients converged after 8k epochs.

4.2.3 Efficient $O(M)$ parameterisation of the approximate posterior

This section demonstrates the effectiveness of our efficient $O(M)$ parameterisation of the approximate posterior $q(\mathbf{u}) = N(\mathbf{u}; \mathbf{K}_{mm}\mu, (\mathbf{K}_{mm}^{-1} + \mathbf{G})^{-1})$. In this section, we refer to sparse variational GP models with our parameterisation using NGD as **our NGD method**. We will first test the performance of **our NGD method** on a simple synthetic GP dataset. We will compare its performance with NGD and Adam using the original approximate posterior parameterised in terms of mean and covariance. We will then, investigate the applicability of **our NGD method** to a real dataset (the NAVAL dataset) as well as the stability of the optimization process in an ill-conditioned setting.

Synthetic Data Experiments: full stochastic optimization

First, we test the performance of **our NGD method** using a synthetic GP detaset in terms of the speed of comvergence and the quality of optimization as well as the output prediction samples. Here we use a sufficient number of inducing inputs so that we can benefit from the computational efficiency of our economical $O(M)$ parameterisation. We assume $M = 50$ is fairly large and appropriate for the synthetic GP dataset. This means that the models with the original parameterisation have $M + M(M+1)/2 = 1325$ variational parameters, and the models with our alternative parameterisation have $2M = 100$ variational parameters, to be optimized by NGD or Adam.

Figure 4.7 shows the process of optimization in terms of the ELBO and hyperparameter values. We can see that, given a sufficient number of inducing inputs, **our NGD method** benefits from computational efficiency and faster convergence without loss of the accuracy at the optimum. Given that the true kernel length scale is 0.1, kernel variance is 0.5, the hyperparameter graphs show that our method finds smaller values for the kernel length scale and kernel variance (discussed in later this section). As for the likelihood variance, all three methods find the true value 0.01.

Figure 4.8 shows the output prediction samples after training. Although all three graphs have similar posterior predictions with almost the same pattern, the top graph of **our NGD method** exhibits a less smooth mean function and confidence bound. This observation agrees with the smaller kernel length scale and kernel variance in Figure 4.7.

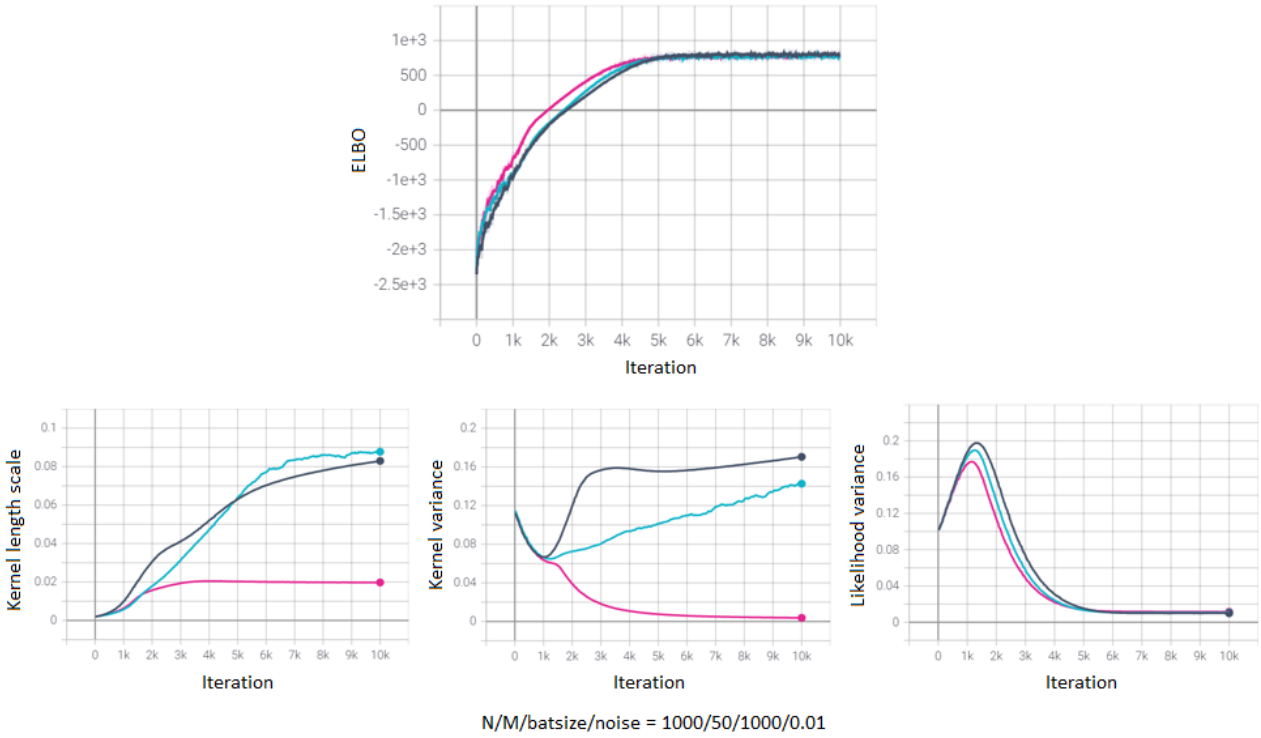


Fig. 4.7 Joint optimization of the hyperparameters and the variational distribution in the case of $N = 1000, M = 50$, no use of minibatch, and noise variance 0.01. **our NGD method** (shown in pink) is faster to converge than both the original NGD (shown in light blue) and Adam (shown in grey), reaching the same lower bound.

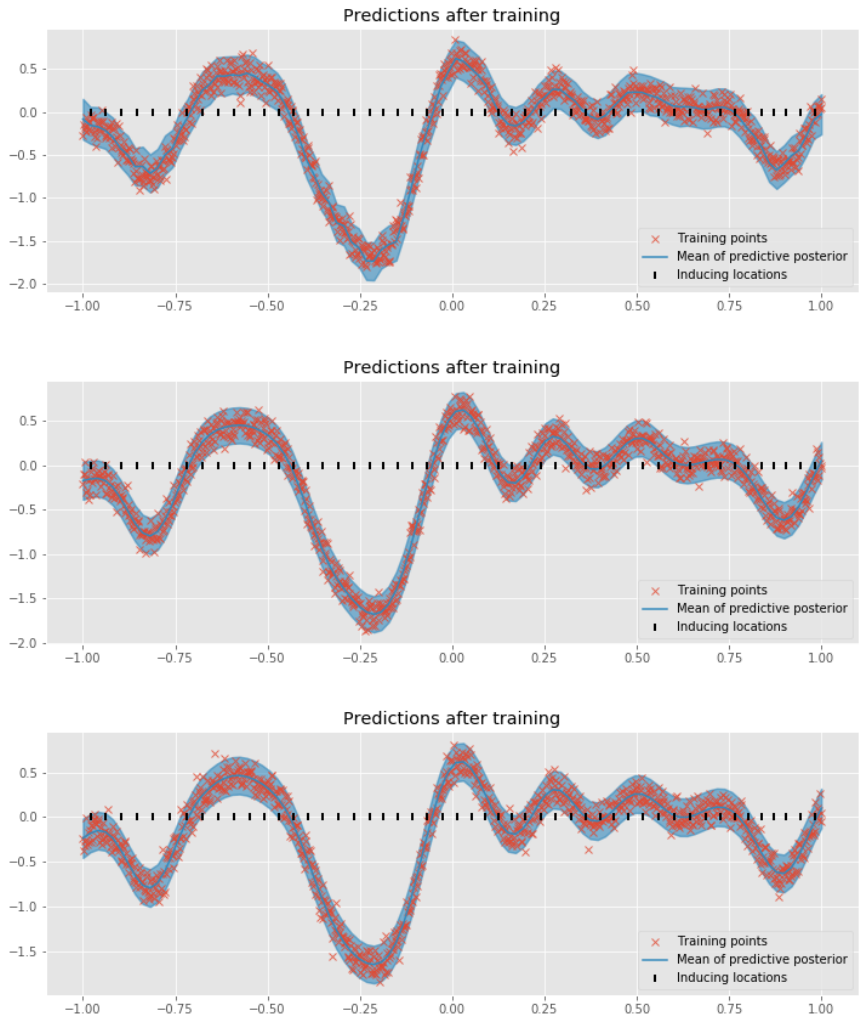


Fig. 4.8 The output prediction samples after training of **our NGD method** (shown in top), Adam (shown in middle), and NGD (shown in bottom).

Real-data: NAVAL Data Experiments

We next consider an ill-conditioned setting. Here we use $M = 100$ inducing inputs so that we can benefit from the computational efficiency of our parameterisation. This means that we have $M + M(M+1)/2 = 5150$ variational parameters for the original model and $2M = 200$ variational parameters for our model.

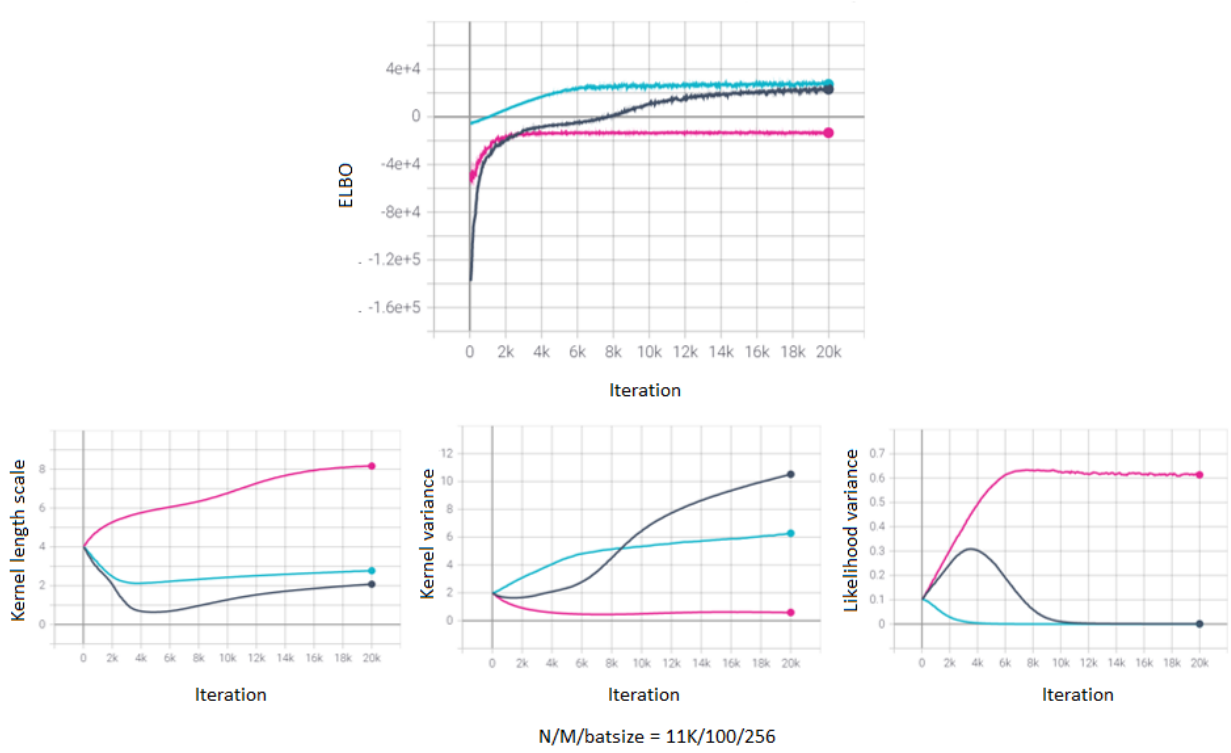


Fig. 4.9 Joint optimization of the hyperparameters and the variational distribution in the NAVAL dataset. NGD (shown in pink) is faster to converge, although it reaches to a larger ELBO. NGD (shown in light blue) has converged second, whereas Adam (shown in grey) cannot converge after quite large number of iterations $20K$.

Figure 4.9 shows that **our NGD method** is not converging to the same optimum that the original methods find, but converging to a lower ELBO. In terms of the speed of convergence, **our NGD method** is the fastest due to the smaller number of parameters and the advantage of the natural gradients. This result agrees with our expectation that our method sacrifices accuracy in favour of a much smaller number of variational parameters. However, it is possible for **our NGD method** to increase the number of inducing inputs without damaging the speed of convergence too much as the number of parameters increases linearly, not quadratically. Note that the computational cost is still $O(M^2)$.

Looking at the hyperparameter optimization process in Figure 4.7 and 4.9, we can see a potential bias towards underestimating the kernel variance. This is because, where the kernel variance is small and the likelihood variance is non zero, the covariance of the approximate posterior S will be very small, and the approximate posterior will essentially be the prior (i.e. our variational parameter $diag(G)$ will be close to 0). The approximate posterior exactly captures the true posterior by having the likelihood terms all go to zero. This makes the KL term small resulting in the bias towards underestimating the kernel variance and potentially overestimating the likelihood variance.

In the all experiments on our efficient $O(M)$ parameterisation, we use a Gaussian likelihood. This ensure the positive definiteness of the matrix $\mathbf{G}$, meaning that we do not need to place constraints upon it. To ensure the positive definiteness in the cases of non-Gaussian likelihoods, it is necessary to place constraints, for example, by using the softplus transformation.

4.3 Discussion

4.3.1 A trade-off in the number of parameters and convexity

In this project and the prior works on the efficient $O(M)$ parameterisation, we seek to have a smaller number of variational parameters (i.e. $2M$ instead of $M + M(M+1)/2$) so that we can benefit from the computationally economical optimization step in terms of memory and the faster convergence. However, there is a known trade-off in the number of parameters. If there are more variables, then the optimisation would be easier in a way because there are a lot to tune. Furthermore, according to (Khan and Nielsen, 2018), the $O(M)$ parameterisation destroys the convexity of the original problem. In their work, a dual decomposition approach is proposed that allows us to reduce the number of parameters while retaining convexity.

Chapter 5

Conclusion

5.1 Summary

This project has investigated the natural gradient method in a sparse variational GP model, finding that the natural gradient method is particularly beneficial when relatively small number of inducing inputs are given. It can find a better or equal optimum within a smaller number of iterations. In practice this is usually the case – the dataset is typically large and high dimensional and the number of inducing inputs is typically set small $M << N$ to benefit from the computational efficiency of the sparse approximation. Even if the number of inducing inputs is relatively large, the natural gradient is at least as good as the Adam optimizer as long as minibatchs are not utilised.

Furthermore, this project has presented a sparse variational GP method which employs an efficient $O(M)$ parameterisaiton for the approximate Gaussian variational posterior together with the natural gradient optimization. As the number of variational parameters is reduced from $M + M(M+1)/2$ to just $2M$, the possibility of achieving the optimal approximate posterior is sacrificed. However, with a fairly large number of inducing inputs, the loss in accuracy may be considered as small. Note that the computational cost still scales quadratically $O(M^2)$ with $O(M)$ parameterisation.

5.2 Further work

We have presented the performance of our sparse variational GP model with an approximate posterior parameterised by $2M$ natural parameters in the affine transformation space. Further investigation into whether the effects of this alternative parameterisation carry over to a wider

variety of settings is crucial, and is likely to yield further insights. Also, the stability of the model in terms of the step size can be investigated.

Our experiments on the natural gradients revealed a number of important effects which were not stressed in the literature, and are indicative that a better understanding of these algorithms is required.

References

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Bui, T. D., Yan, J., and Turner, R. E. (2017). A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *The Journal of Machine Learning Research*, 18(1):3649–3720.

Crisher, B. and Souva, M. (2013). Power At Sea: A Naval Dataset, 1865-2011.

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

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR.

Hensman, J., Rattray, M., and Lawrence, N. D. (2012). Fast variational inference in the conjugate exponential family. *arXiv preprint arXiv:1206.5162*.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(5).

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

Khan, M. E. and Nielsen, D. (2018). Fast yet simple natural-gradient descent for variational inference in complex models. In *2018 International Symposium on Information Theory and Its Applications (ISITA)*, pages 31–35. IEEE.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In *Proceedings of the 16th annual conference on neural information processing systems*, number CONF, pages 609–616.

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

Matthews, A. G. d. G., Hensman, J., Turner, R. E., and Ghahramani, Z. (2015). On sparse variational methods and the kullback-leibler divergence between stochastic processes. *arXiv preprint arXiv:1504.07027*.

Matthews, A. G. d. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). Gpflow: A gaussian process library using tensorflow. *J. Mach. Learn. Res.*, 18(40):1–6.

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792.

Panos, A., Dellaportas, P., and Titsias, M. K. (2018). Fully scalable gaussian processes using subspace inducing inputs. *arXiv preprint arXiv:1807.02537*.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

Salimbeni, H., Eleftheriadis, S., and Hensman, J. (2018). Natural gradients in practice: Non-conjugate variational inference in gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 689–697. PMLR.

Sato, M.-A. (2001). Online model selection based on the variational bayes. *Neural computation*, 13(7):1649–1681.

Seeger, M. W., Williams, C. K., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*, pages 254–261. PMLR.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257.

Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. (2009). Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR.

Ulapane, N., Thiyagarajan, K., and Kodagoda, S. (2020). Hyper-parameter initialization for squared exponential kernel-based gaussian process regression. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1154–1159. IEEE.

Appendix A

Supplementary material for VFE

Reinterpretation of the proof for VFE

Matthews et al. (2015) gave a mathematically rigorous treatment of the VFE framework. It generalized the framework, showing that marginal consistency of augmentation is not enough to guarantee consistency of variational inference with the original model. Here, we briefly introduce their proof.

Let us assume the input space $\mathbb{R}$ is finite and can be partitioned into three disjoint subspaces: a set of inducing inputs $\mathbf{Z}$ of size M, the set of input positions for the observed data $\mathbf{X}$ of size N, and the rest of the index set $\mathbf{X}_{\#}$, so that $\mathbb{R} \equiv \mathbf{Z} \cup \mathbf{X} \cup \mathbf{X}_{\#}$. The function of interest f maps the index set to sets of function values: $f_{\mathbf{Z}}, f_{\mathbf{X}}$ and $f_{\#}$. The set of observation corresponding to the input set $\mathbf{X}$ is $\mathbf{Y}$. The variational distribution at those data points is taken to have the form:

$$q(f) = p(f_{\#}, f_{\mathbf{X}} \mid f_{\mathbf{Z}})\, q(f_{\mathbf{Z}}) \tag{A.1}$$

Then, the KL divergence between the approximating and posterior processes (i.e. between the variational distribution and the full posterior distribution) is given as follows:

$$\begin{aligned} KL(q(f)\,||p(f|\mathbf{Y})) &= KL(q(f_{\mathbf{X}}, f_{\mathbf{Z}}, f_{\#})\,||p(f_{\mathbf{X}}, f_{\mathbf{Z}}, f_{\#}|\mathbf{Y})) && \text{(A.2)} \\ &= \int q(f_{\mathbf{X}}, f_{\mathbf{Z}}, f_{\#}) \log \left\{ \frac{q(f_{\mathbf{X}}, f_{\mathbf{Z}}, f_{\#})}{p(f_{\mathbf{X}}, f_{\mathbf{Z}}, f_{\#} \mid Y)} \right\} df_{\#} df_{\mathbf{X}} df_{\mathbf{Z}} && \text{(A.3)} \\ &= \int q(f_{\mathbf{X}}, f_{\mathbf{Z}}) \log \left\{ \frac{q(f_{\mathbf{X}}, f_{\mathbf{Z}})}{p(f_{\mathbf{X}}, f_{\mathbf{Z}} \mid Y)} \right\} df_{\mathbf{X}} df_{\mathbf{Z}} && \text{(A.4)} \end{aligned}$$

where more detailed derivation can be found in section 2 in (Matthews et al., 2015). The last line does not contain $f_{\#}$, meaning that all the other function values $f_{\#}$ marginalize and we only need to keep track of the distribution over function values $f_{\mathbf{X}}$ and $f_{\mathbf{Z}}$.

However, if the input space $\mathbb{R}$ is infinite, this is not the case since the infinite-dimensional integral is required to compute the KL divergence. We will not go into details here, but it will require an alternative proof for infinite index sets shown in (Matthews et al., 2015). In practice, we suppose that we are dealing with a GP valued only on a sufficiently large finite subset of infinite input space $\mathbb{R}$, so that we can benefit from this VFE framework.