Scalable Inference and Private Co-training for Gaussian Processes

Owen Thomas
St Hugh's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2017

For my family.

Acknowledgements

Gratitude must be expressed to my supervisor Professor Chris Holmes, my collaborator Dr. Michalis Titsias, and to the faculty, researchers, staff and fellow students who have helped with the development of this research. Further thanks go to my family and friends for their support, encouragement and diversion while this work was being done.

Abstract

Two principal problems are pursued in this thesis: that of scaling inference for Gaussian process regression to very large numbers of data points, and that of differentially private co-training between multiple Gaussian processes with distinct private views of the data.

The first chapter acts as an introduction to Bayesian nonparametric regression and standard techniques for performing scalable inference and differentially private communication with Gaussian Processes.

The second chapter explores the use of Tucker decomposition and Kronecker structure of variational distributions in order to use very large numbers of inducing points for scalable variational Gaussian processes. The methods use a massive number of gridded inducing points, enabling more expansive variational approximations, but the computational cost of the methods developed scales sublinearly with m, the number of inducing points used. Computational experiments are performed and compared with standard variational Gaussian processes, and the new method is demonstrated to be more efficient for data sets of moderate dimensionality.

The third chapter pursues the use of stochastic algorithms for evaluating unbiased approximations to the gradients of the lower bound for a scalable variational Gaussian process. Consequently, $O(m^2)$ inference is possible, in contrast to the $O(m^3)$ inference for a standard variational Gaussian process. Systematic computational experiments are performed and compared with a standard variational Gaussian process, with mixed results.

The fourth chapter considers the problem of preserving privacy when engaging in co-training for regression with multiple Gaussian processes, each trained on different sets of covariates or "views". Information is exchanged between views with predictions on unlabelled data: various methods are introduced to incorporate the predictive distributions, and mechanisms for formally preserving the differential privacy of the response variable are included. Analogies are drawn between the Kronecker product across dimensions of the second chapter and the fourth chapter's co-training between views. Experiments are performed on simulated and real data, and the results are presented and analysed. It is shown that differentially private exchange of information between views via predictions on unlabelled data can improve the performance of the models.

The fifth chapter concludes.

Contents

List of Figures

Chapter 1

Introduction

Statisics and data-driven modelling are growing ever more important in our lives. The profusion of data available across almost every field of study, be it from the medical, social or natural sciences, poses a combination of challenges and opportunites for statisticians and machine learning practitioners. There exist enormous practical benefits to having quantitative models for decision making or data interpretation, but the problems of computational scalability and data privacy consistently arise in many practical circumstances. In this thesis, I attempt to address these two problems for the Gaussian Process (GP), a Bayesian nonparametric model frequently used for various tasks in supervised and unsupervised learning.

In this chapter, I introduce the fundamentals of Bayesian modelling in Section 1.1, building up to a definition of a Gaussian Process in Section 1.2, and then covering the existing literature on scalability and privacy for GPs in Sections 1.3 and 1.4, respectively.

1.1 Bayesian Learning

The problem of Bayesian learning concerns itself with deriving posterior belief distributions over parameters of interest, given observed data. The process of inference for a parameter θ begins with a statement of $p(\theta)$, the prior belief distribution of the statistician. The posterior distribution $p(\theta|\mathcal{D})$ of θ given the data $\mathcal{D}$ can be derived using Bayes' theorem by multiplying the prior with the data likelihood $p(\mathcal{D}|\theta)$ and normalising with the model evidence $p(\mathcal{D})$:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{1.1}$$

The posterior distribution can be used to characterise expected utilities and facilitate decision making: Bayes' theorem can be derived from decision theory as an optimal belief update (Bernardo and Smith, 2001).

A visual example of a Bayesian update can be seen in Figure 1.1.

Given a sufficient amount of data, the posterior distribution can converge to the true value of θ. However, this assumes an accurate specification of the model, such that the true data generating process is given non-zero weight by the prior. In cases of misspecification, the

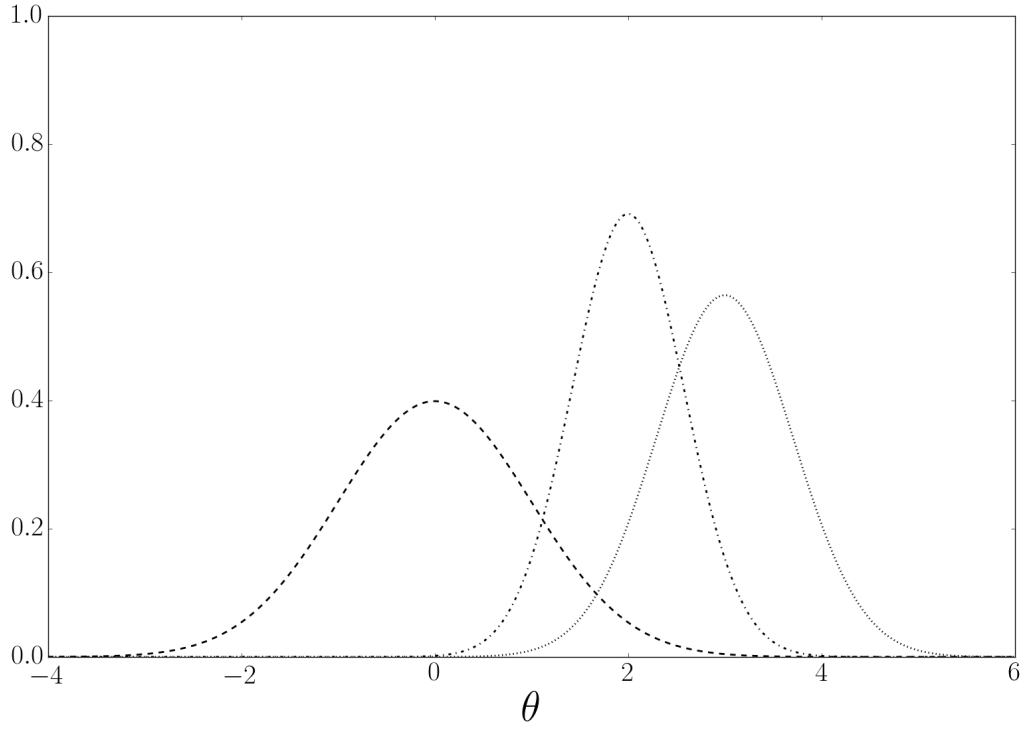


Figure 1.1: A Bayesian update for a parameter θ. The prior $p(\theta)$ (dashed line) is multiplied with the data likelihood $p(\mathcal{D}|\theta)$ (dotted line) and then normalised to give the parameter posterior $p(\theta|\mathcal{D})$ (dot-dashed line). The Gaussian prior is conjugate to the Gaussian likelihood, resulting in an analytically tractable Gaussian posterior.

posterior converges to the point in the prior closest in Kullback-Leibler divergence to the true model, under mild regularity assumptions (Grünwald, 2012).

Bayesian computation is often challenging: the posterior can be impossible to access analytically, often due to the computation of the normalising factor $p(\mathcal{D})$, frequently called the "evidence" or the "marginal likelihood" (ML):

$$p(\mathcal{D}) = \int_{\theta^*} p(\mathcal{D}|\theta^*)p(\theta^*)d\theta^*. \tag{1.2}$$

Approximate Bayesian inference is a vast and complex field. The most popular approaches for characterising the posterior include Laplace approximations, Monte Carlo methods and Variational inference.

1.1.1 Laplace Approximations

A Laplace approximation can be performed by optimising the posterior distribution to find the *maximum a posteriori* (MAP) estimate $\hat{\theta}_{MAP} = \arg\max p(\theta|\mathcal{D})$ for the parameter. This is sometimes used by itself to represent an optimal estimate for the parameter, but for a full Laplace approximation, the second derivative of the posterior distribution with respect to θ, $p''(\theta|\mathcal{D})$, is then evaluated at $\hat{\theta}_{MAP}$ to return a characterisation of the width of the posterior distribution. The MAP estimate and MAP second derivative are then used to define a Gaussian distribution which can be used as a proxy for the true posterior.

The reliance of this method on an optimisation makes it computationally appealing, as convergence to a mode of the posterior is generally faster than characterising the entire distribution. However, this introduces vulnerabilities to misrepresenting multimodal posterior distributions. Similarly, use of information local to the MAP estimate might lead it to misrepresent a significantly non-Gaussian posterior, and it is completely unable to represent any skewness in the distribution. However, for certain models it can represent a very powerful and computationally appealing method if the model posterior's properties can be adequately captured by the approximation (Blangiardo et al., 2013).

1.1.2 Monte Carlo Sampling

Monte Carlo methods aim to generate a collection of samples from the posterior distribution, used to act as a discrete representation of the continuous distribution (Robert, 2004). In principle such methods should be able to characterise any posterior distribution, as they do not make distributional assumptions about the posterior, although in practice this depends on how well the stochastic algorithm converges or "mixes". Reducing the variance of the generated samples is one of the chief concerns for this family of methods.

Monte Carlo sampling will often demonstrate an error asymptotically scaling as $O(\frac{1}{\sqrt{N}})$ with number of samples N. Such provable convergence diagnostics are often very useful, although it is not always clear that the asymptotic limit is truly reached in practice. The inverse square root scaling is also not very appealing: a well-mixed Monte Carlo solution can frequently be considered an accurate but expensive option.

1.1.3 Variational Inference

A further method for approximately characterising a Bayesian posterior is that of variational inference (Blei et al., 2017). Within this framework, an analytically accessible distribution is introduced to represent the posterior. A lower bound to the marginal likelihood of the data $p(\mathcal{D})$ is then derived, which is used as a target function to optimise the parameters of the model and variational distribution. Such an optimisation corresponds to minimising the Kullback-Leibler distance between the variational distribution and the true posterior.

Variational methods have a strong connection to Expectation-Maximisation (EM) methods, which derive proxy posterior distributions for some variables and point estimates for others. By contrast, a fully variational method derives analytically realisable approximate posteriors for variables of interest, and also a lower bound to the marginal likelihood, allowing for principled model comparison. However, the method's reliance on optimisation renders it vulnerable to local minima and poor convergence diagnostics, and the derivation of the variational bound and updates can be challenging. Further, the use of explicit distributions in place of the true posterior introduces potentially flawed assumptions of how to capture the posterior uncertainty.

1.2 Supervised Learning and Regression

One of the canonical problems in statistics is that of regression, in which a continuous response variable with n observations $\mathbf{y}$ is predicted using a corresponding observed n by p matrix of covariates $\mathbf{X}$. Formally, we would like to characterise the conditional distribution $p(\mathbf{y}|\mathbf{X})$, and the predictive distribution $p(\mathbf{y}^*|\mathbf{X}, \mathbf{y}, \mathbf{X}^*)$ of the response variable $\mathbf{y}^*$ corresponding to the covariates $\mathbf{X}^*$.

Within a Bayesian framework, it would be standard to introduce the model parameters or latent variables θ to define a model likelihood $p(\mathbf{y}|\mathbf{X}, \theta)$ and predictive distribution $p(\mathbf{y}^*|\mathbf{X}, \mathbf{y}, \mathbf{X}^*, \theta)$. The model posterior $p(\theta|\mathbf{y}, \mathbf{X})$ is then learned and the parameters θ are integrated out, in order to give the most accurate estimation of predictive means and variances.

1.2.1 Linear Regression

The simplest non-trivial Bayesian regression model is Bayesian linear regression, in which the response variable is assumed to be a linear sum of the covariates with some additional Gaussian noise. Formally speaking, the model for the conditional dependence can be written:

$$f_i = \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{X}_{ij} = \boldsymbol{\beta}^T \mathbf{x}_i, \tag{1.3}$$

$$y_i = f_i + \epsilon_i, \tag{1.4}$$

where f_i is a latent variable formed by the linear combination of covariates, $\boldsymbol{\beta}$ is a vector of the coefficients β_j and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise added to produce the response variable. The model parameters are the coefficients $\boldsymbol{\beta}$ and noise parameter σ^2. The corresponding model likelihood for these parameters $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \sigma^2)$ is:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \sigma^2) \propto (\sigma^2)^{-n/2} \exp\Big(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\Big). \tag{1.5}$$

To perform Bayesian inference over the parameters $\boldsymbol{\beta}$, it is possible to introduce Gaussian priors $p(\boldsymbol{\beta})$, and consequently an implicit prior over the latent variable $\mathbf{f}$:

$$p(\boldsymbol{\beta}) = \mathcal{N}(\mu, \Sigma), \tag{1.6}$$

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{X}\mu, \mathbf{X}\Sigma\mathbf{X}^T). \tag{1.7}$$

We notice that we can derive an expression for the log marginal likelihood (ML) of the data $p(\mathbf{y}|\mathbf{X}, \sigma^2)$ directly from the statement of the prior:

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}, \sigma^2) &= \log \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} \\ &= -\frac{1}{2}(\mathbf{y} - \mathbf{X}\mu)^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \mathbf{X}\mu) \\ &\quad - \frac{1}{2}\log|\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I}| - \frac{n}{2}\log 2\pi. \end{aligned} \tag{1.8}$$

We can then use Bayes' Theorem to derive posteriors over $\boldsymbol{\beta}$ and $\mathbf{f}$ given the response variable:

$$\begin{aligned} p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \sigma^2) = \mathcal{N}\Big(&\mu + \Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \mathbf{X}\mu), \\ &\Sigma - \Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}\mathbf{X}\Sigma\Big), \end{aligned} \tag{1.9}$$

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}, \mathbf{X}, \sigma^2) = \mathcal{N}\Big(&\mathbf{X}\mu + \mathbf{X}\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \mathbf{X}\mu), \\ &\mathbf{X}\Sigma\mathbf{X}^T - \mathbf{X}\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}\mathbf{X}\Sigma\mathbf{X}^T\Big). \end{aligned} \tag{1.10}$$

These posteriors can be used to perform predictions or evaluate expectations of utilities. To predict $\mathbf{f}^*$ and $\mathbf{y}^*$ at a given covariate $\mathbf{x}^*$, we can use the following predictive distributions:

$$\begin{aligned} p(\mathbf{f}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*, \sigma^2) = \mathcal{N}\Big(&\mathbf{x}^*\mu + \mathbf{x}^*\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \mathbf{X}\mu), \\ &\mathbf{x}^*\Sigma\mathbf{x}^{*T} - \mathbf{x}^*\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}\mathbf{X}\Sigma\mathbf{x}^{*T}\Big), \end{aligned} \tag{1.11}$$

$$\begin{aligned} p(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*, \sigma^2) &= \int p(\mathbf{y}^*|\mathbf{f}^*, \sigma^2)p(\mathbf{f}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*, \sigma^2)d\mathbf{f}^* \\ &= \mathcal{N}\Big(\mathbf{x}^*\mu + \mathbf{x}^*\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \mathbf{X}\mu), \\ &\qquad \mathbf{x}^*\Sigma\mathbf{x}^{*T} - \mathbf{x}^*\Sigma\mathbf{X}^T(\mathbf{X}\Sigma\mathbf{X}^T + \sigma^2\mathbb{I})^{-1}\mathbf{X}\Sigma\mathbf{x}^{*T} + \sigma^2\Big). \end{aligned} \tag{1.12}$$

We observe that the posteriors and predictives are analytically accessible, with an $O(n^3)$ solution of a linear system and an $O(np^2)$ matrix product. An example of a Bayesian linear model fitted on a simple regression problem can be seen in Figure 1.2.

We also notice that all of the posteriors and predictives are still conditioned on σ^2. It is possible to put a conjugate inverse gamma prior over σ^2, but the subsequent analytical posteriors are much less elegant, so it is common to use Maximum Likelihood estimates or Gibbs sampling instead. In the examples presented in Figures 1.2, 1.3, 1.4 and 1.5, we perform prediction with the Maximum Likelihood estimates of σ^2.

1.2.2 Nonlinear Regression

We have successfully demonstrated how to perform a Bayesian analysis of a linear regression model. However, we began the derivation with an assumption that the response variable is formed by a linear sum of the covariates: this severely restricts the set of models under consideration, and increases the probability of model misspecification.

Consequently, it is of interest to introduce nonlinearities into the model in order to increase its expressive capacity. A clear expansion would be to suggest that the latent variable $\mathbf{f}$ is formed from a sum of q known nonlinear functions or "features" $\phi_j(\mathbf{X})$ of the covariates $\mathbf{X}$. Common choices for these include polynomial functions, sinusoids or radial basis functions. The following model is thus pursued:

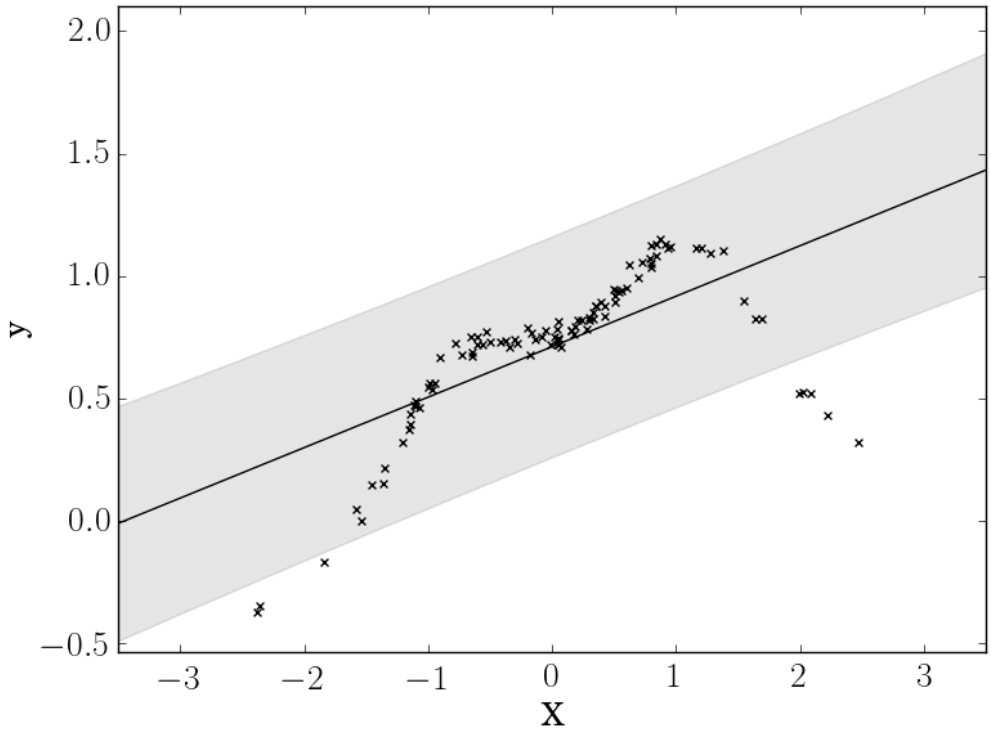


Figure 1.2: A plot of a one-dimensional simulated regression data set, shown with the posterior of a linear model. The line displayed is the posterior mean predictive and the 95% credible interval is shaded.

$$f_i = \beta_0 + \sum_{j=1}^{q} \beta_j \phi_j(\mathbf{x}_i) = \boldsymbol{\beta}^T \Phi(\mathbf{x}_i), \tag{1.13}$$

$$y_i = f_i + \epsilon_i, \tag{1.14}$$

where $\Phi(\mathbf{x}_i)$ is the vector of functions associated with data point i. The analysis above can then be repeated, replacing the n by p data matrix $\mathbf{X}$ with the n by q data feature matrix $\Phi_{\mathbf{X}}$, producing the following log marginal likelihood and predictive distributions:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X}, \sigma^2) = &-\frac{1}{2}(\mathbf{y} - \Phi_{\mathbf{X}}\mu)^T(\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{X}}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \Phi_{\mathbf{X}}\mu) \\ &-\frac{1}{2}\log|\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{X}}^T + \sigma^2\mathbb{I}| - \frac{n}{2}\log 2\pi, \end{aligned} \tag{1.15}$$

$$\begin{aligned}p(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*, \sigma^2) = \mathcal{N}\Big(&\Phi_{\mathbf{x}^*}\mu + \Phi_{\mathbf{x}^*}\Sigma\Phi_{\mathbf{X}}^T(\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{X}}^T + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \Phi_{\mathbf{X}}\mu), \\ &\Phi_{\mathbf{x}^*}\Sigma\Phi_{\mathbf{x}^*}^T - \Phi_{\mathbf{x}^*}\Sigma\Phi_{\mathbf{X}}^T(\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{X}}^T + \sigma^2\mathbb{I})^{-1}\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{x}^*}^T + \sigma^2\Big). \end{aligned} \tag{1.16}$$

We see that the log ML and predictive distributions remain tractable, requiring $O(n^3)$, $O(n^2q)$ and $O(nq^2)$ computations. The question remains how to choose which basis functions to use: in principle the marginal likelihood $p(\mathbf{y}|\mathbf{X}, \sigma^2)$ can be used to perform model comparison or optimisation of different basis functions. However, it would be desirable for a model to not commit *a priori* to a particular set of basis functions in specific locations.

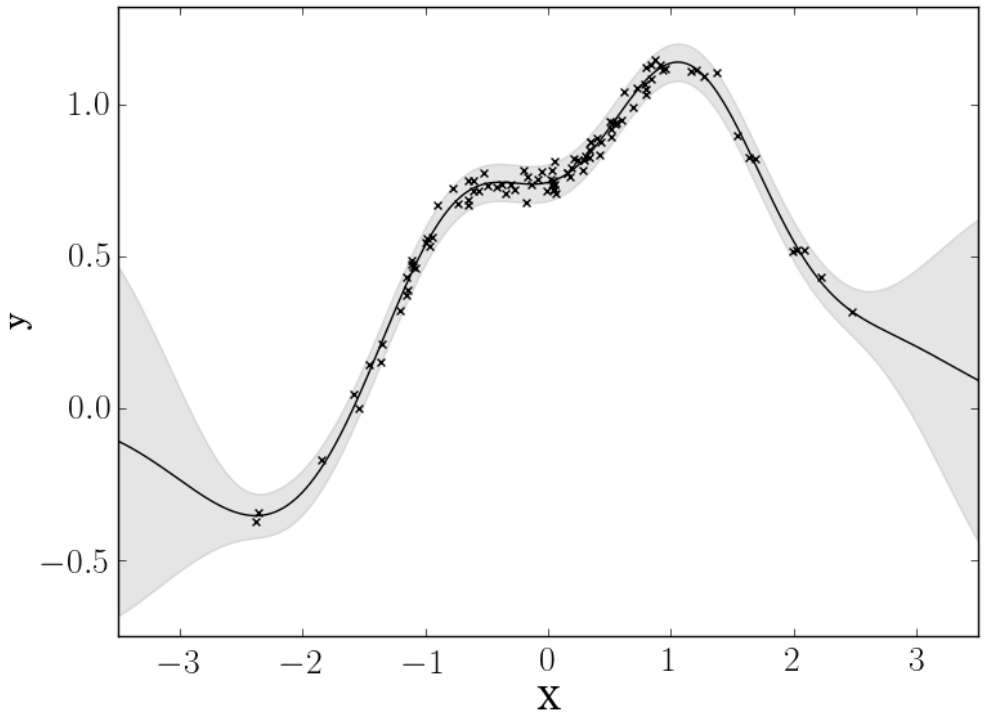


Figure 1.3: A plot of a one-dimensional simulated regression data set, shown with the posterior of a finite basis with 10 radial basis functions. The lines display is the posterior mean predictive and the 95% credible interval is shaded.

An example of a finite basis model with radial basis functions run on a simple regression problem can be seen in Figure 1.3.

1.2.3 Gaussian Processes and Nonparametric Regression

A solution to the problem of specifying basis functions might be to massively increase their number: once the matrix $\Phi_{\mathbf{X}}\Sigma\Phi_{\mathbf{X}}^T$ has been computed, then the cost of evaluating the marginal likelihood and predictive distribution is independent of q. Further, if the prior covariance Σ is diagonal, then the computation of this matrix is $O(qn^2)$, so the addition of new basis functions exhibits cheap linear scaling.

For certain classes of basis functions then a "kernel trick" can be performed, in which the number of basis functions q is taken to infinity, such that the inner product between feature vectors for data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is replaced with an analytical covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. Consequently, the covariance function can be evaluated without explicit reference to the basis functions used and without computing the inner product.

The implicit use of an infinite number of uniformly populated basis functions ameliorates the *a priori* commitment to specifying basis functions before seeing the data. The kernel-driven method is able to adapt to new data and implicitly place basis functions at any potential location, and as such represents a very flexible model. The presence of an infinite-dimensional set of basis functions with a finite-dimensional representation in the form of the covariance

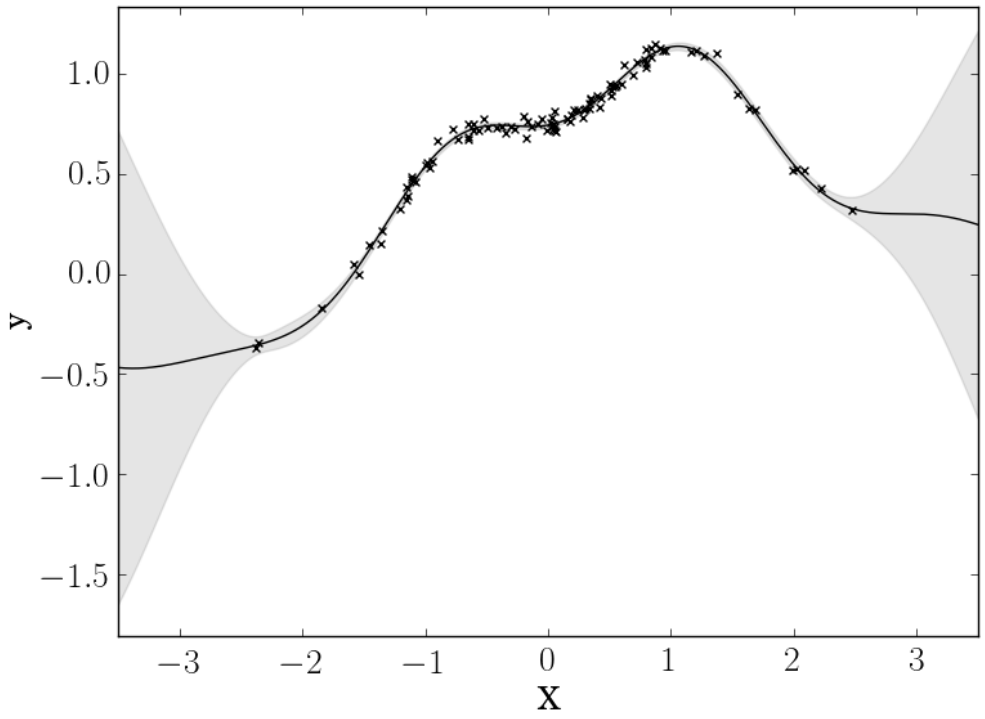


Figure 1.4: A plot of a one-dimensional simulated regression data set, shown with the posterior of a Gaussian Process with an exponentiated quadratic kernel. The line displayed is the posterior mean predictive and the 95% credible interval is shaded.

function is characteristic of a "nonparametric" method, in contrast to the "parametric" finite-basis models examined previously.

If we restate the marginal likelihoods and predictive distributions from the parametric models within the covariance-driven formalism, and set the prior μ to zero without loss of generality, we achieve some canonical expressions for a Gaussian Process (GP) (Rasmussen and Williams):

$$\log p(\mathbf{y}|\mathbf{X},\theta,\sigma^2) = -\frac{1}{2}\mathbf{y}^T(\mathbf{K_{ff}}+\sigma^2\mathbb{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K_{ff}}+\sigma^2\mathbb{I}| - \frac{n}{2}\log 2\pi, \tag{1.17}$$

$$p(\mathbf{y}^*|\mathbf{y},\mathbf{X},\mathbf{x}^*,\theta,\sigma^2) = \mathcal{N}\Big(\mathbf{k_{f^*f}}(\mathbf{K_{ff}}+\sigma^2\mathbb{I})^{-1}\mathbf{y}, k_{\mathbf{f^*f^*}} - \mathbf{k_{f^*f}}(\mathbf{K_{ff}}+\sigma^2\mathbb{I})^{-1}\mathbf{k_{ff^*}} + \sigma^2\Big), \tag{1.18}$$

where θ is a "hyperparameter" of the covariance function, $\mathbf{K_{ff}}$ is an n by n matrix of the covariance function evaluated between all the covariates $\mathbf{X}$, $\mathbf{k_{f^*f}}$ is a vector of the covariance function evaluated between the predictive covariate $\mathbf{x}^*$ and covariates $\mathbf{X}$, and $k_{\mathbf{f^*f^*}}$ is the covariance function evaluated between the predictive covariate $\mathbf{x}^*$ and itself.

Equations 1.17 and 1.18 correspond to a Gaussian latent variable model, with a latent variable $\mathbf{f}$ with prior mean of zero and covariance $\mathbf{K_{ff}}$. This is combined with an independent multivariate Gaussian likelihood, with the latent variable $\mathbf{f}$ having been analytically marginalised out i.e.:

$$p(\mathbf{f}|\mathbf{X},\theta) = \mathcal{N}(\mathbf{f}|0,\mathbf{K_{ff}}), \tag{1.19}$$

$$p(\mathbf{y}|\mathbf{f},\sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{f},\sigma^2\mathbb{I}), \tag{1.20}$$

$$p(\mathbf{y}|\mathbf{X},\theta,\sigma^2) = \mathcal{N}(\mathbf{y}|0,\mathbf{K_{ff}}+\sigma^2\mathbb{I}). \tag{1.21}$$

We can see an example of a Gaussian process fitted to a simple regression problem in Figure 1.4. The credible interval is smaller than that of the finite basis model, as the more general nonparametric model can converge on the true data generating process used in this example.

It should be made clear that only certain covariance functions are admissible: the matrix $\mathbf{K_{ff}}$ is the model's representation for the covariance matrix of the latent variable $\mathbf{f}$ evaluated at the data. Consequently, we are constrained to using covariance functions that are guaranteed to generate positive definite covariance matrices: such functions are themselves known as positive definite.

Many valid covariance functions exist: some of the most commonly used are displayed in Table 1.1.

Kernel name	Function $k(\mathbf{x}_i,\mathbf{x}_j,\theta)$	Hyperparameters θ
Exponentiated Quadratic	$\exp(-\frac{r^2}{2l^2})$	l
Rational Quadratic	$(1+\frac{r^2}{2\alpha l^2})^{-\alpha}$	l,α
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)}(\frac{\sqrt{2\nu}}{l}r)^\nu K_\nu(\frac{\sqrt{2\nu}}{l}r)$	l,ν
Exponential	$\exp(-\frac{r}{l})$	l
Linear	$\sum_d \sigma_d^2(\mathbf{x}_{id}-c)(\mathbf{x}_{jd}-c)$	σ_d^2, c
Constant	σ^2	σ^2
Periodic	$\exp(-\frac{2\sin^2(\pi r/p)}{l^2})$	l,p

Table 1.1: Some of the most common covariance functions evaluated between data points $\mathbf{x}_i$ and $\mathbf{x}_j$. The functions written in terms of $r=|\mathbf{x}_i-\mathbf{x}_j|$ are "stationary" covariance functions. Γ is the gamma function, and K_ν is the modified Bessel function of the second kind.

It is also possible to generate new positive definite functions by combining existing ones, with the most common operations being a sum or product of the function evaluations. For multidimensional data, it is common to use "separable" covariance functions, in which a multidimensional kernel is built by taking a product across multiple one-dimensional kernels. This introduces the possibility of associating a different length scale with each dimension, enabling Automatic Relevance Determination (ARD) to learn which dimensions are powerful predictors.

Each of the covariance functions mentioned are described by hyperparameters, often representing length scales, periodicities or kernel powers. These need to be learned for good model performance. A fully Bayesian approach would characterise the posterior over the hyperparameters given the data, then integrate them out. However, we can use optimised MAP estimates for the hyperparameters and still claim to be performing Bayesian analysis as we are integrating out all of the uncertainty in the latent variable $\mathbf{f}$ for a given value of θ.

Gaussian Processes have some potential disadvantages compared to parametric models: there can be a loss of interpretability when moving from explicit basis functions to covariance

functions, and it is possible that the very broad prior implicit in a GP is not always appropriate. GPs also contain a strong smoothness assumption brought about by the infinite basis function density limit, so often struggle to model discontinuities in data. However, if we have prior information that motivates the use of a specific mean function, it is possible to run a semiparametric model which combines an explicit mean function and a covariance function.

The model presented above assumed homoscedastic Gaussian noise relating the latent variable $\mathbf{f}$ to the observed response data $\mathbf{y}$. Work has been done to generalise this dependence, with heteroscedastic or t-distributed noise used (Titsias and Lázaro-Gredilla, 2011; Jylänki et al., 2011). Similarly, it is possible to use a sigmoid likelihood to use Gaussian Processes for classification of discrete data (Nickisch and Rasmussen, 2008). However, the use of non-Gaussian likelihoods disturbs the conjugacy of the latent variable and the likelihood, making the posterior inference much more challenging.

1.3 Scalability of Gaussian Processes

The computational cost of evaluating the marginal likelihood and predictive distribution is $O(n^3)$ for the matrix decomposition, and the evaluation of the covariance matrix $\mathbf{K_{ff}}$ is $O(pn^2)$ for most common covariance functions. The quite strong cubic scaling means that running inference or prediction for an exact, unstructured GP on a desktop computer is limited to about $n = 2,000$. The $O(n^2)$ memory demands also represent a significant challenge for many widely used computational resources.

1.3.1 Inducing Points and Variational Inference

A well established method for scaling Gaussian Processes to large n is that of rank reduction, in which the full covariance matrix is replaced with a low-rank approximation of size m, resulting in $O(nm^2)$ complexity and $O(nm)$ memory demands, with $m < n$ (Quiñonero-Candela and Rasmussen, 2005). There are principled variational Bayesian interpretations of this method, initially developed in (Titsias, 2009). A version of the derivation following (Hensman et al., 2013) is presented below. Another insightful tutorial for the variational inference can be found in (Gal and van der Wilk, 2014).

The latent variable $\mathbf{f}$, as defined over n data points, can be projected down to another Gaussian variable $\mathbf{u}$ defined over $m < n$ "inducing points" using the covariance matrix $\mathbf{K_{fu}}$ and standard Gaussian identities:

$$p(\mathbf{y}|\mathbf{f}, \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbb{I}), \tag{1.22}$$

$$p(\mathbf{f}|\mathbf{u}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \tilde{\mathbf{K}}), \tag{1.23}$$

$$p(\mathbf{u}|\theta) = \mathcal{N}(\mathbf{u}|0, \mathbf{K_{uu}}), \tag{1.24}$$

where $\tilde{\mathbf{K}} = \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}$. We can apply Jensen's inequality to $\log p(\mathbf{y}|\mathbf{u}, \theta, \sigma^2)$:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{u},\theta,\sigma^2) &= \log \int p(\mathbf{f}|\mathbf{u},\theta)p(\mathbf{y}|\mathbf{f},\sigma^2)d\mathbf{f} \\ &\geq \int p(\mathbf{f}|\mathbf{u},\theta)\log p(\mathbf{y}|\mathbf{f},\sigma^2)d\mathbf{f} = \mathcal{L}_1. \end{aligned} \tag{1.25}$$

Further, if the likelihood $p(\mathbf{y}|\mathbf{f},\sigma^2)$ can be factorised across the data points, then the variational bound $p(\mathbf{y}|\mathbf{u},\theta,\sigma^2)$ can also be factorised:

$$p(\mathbf{y}|\mathbf{u},\theta,\sigma^2) \geq \exp(\mathcal{L}_1) = \prod_{i=1}^{n} \mathcal{N}(y_i|\mu_i,\sigma^2)\exp(-\frac{1}{2}\tilde{\mathbf{k}}_{ii}\sigma^{-2}), \tag{1.26}$$

where $\boldsymbol{\mu} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}$. We are now in a position to calculate a variational lower bound on the posterior $p(\mathbf{y}|\mathbf{X},\theta,\sigma^2)$ by marginalising over the latent variable at the inducing points:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X},\theta,\sigma^2) &= \log \int p(\mathbf{y}|\mathbf{u},\theta)p(\mathbf{u}|\theta)d\mathbf{u} \\ &\geq \log \int \exp(\mathcal{L}_1)p(\mathbf{u}|\theta)d\mathbf{u} \\ &= \log \mathcal{N}(\mathbf{y}|0,\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}} + \sigma^2\mathbb{I}) - \frac{1}{2\sigma^2}\mathrm{Tr}(\tilde{\mathbf{K}}) = \mathcal{L}_2. \end{aligned} \tag{1.27}$$

The bound $\mathcal{L}_2$ can be used to learn the hyperparameters of the kernel in place of the exact GP marginal likelihood. We see that this bound is no longer factorisable across the data, but now contains no operations greater than linear in n, with general complexity $O(nm^2)$ and memory demands $O(mn)$. This representation introduces the presence of m inducing points, the locations of which might need to be optimised. Since this introduces an extra mp variables to the model learning stage, it is sometimes more convenient to instead use an *a priori* principled subset of the data points without optimisation (Seeger et al., 2003).

The components of the bound $\mathcal{L}_2$ have an interesting interpretation: the first term minimises the predictive error of the model on the observed data, and the second term $-\frac{1}{2\sigma^2}\mathrm{Tr}(\tilde{\mathbf{K}})$ ensures that the variational approximation remains close to the exact GP distribution.

A variational GP with 10 inducing points trained on a simple regression problem can be seen in Figure 1.5.

It is valuable to observe that the only inversion is of $O(m^3)$, and the $O(n)$ operations now consist of multiplication and summation, such that they can be distributed across multiple processors effectively. This was pursued in (Gal et al., 2014) and (Dai et al., 2014).

1.3.2 Stochastic Variational Inference

Further reduction in computational cost can be achieved by performing another variational approximation which allows the variational bound to be factorised over the data points. At this point, the log variational bound becomes a sum over the training data, such that stochastic variational inference can be performed.

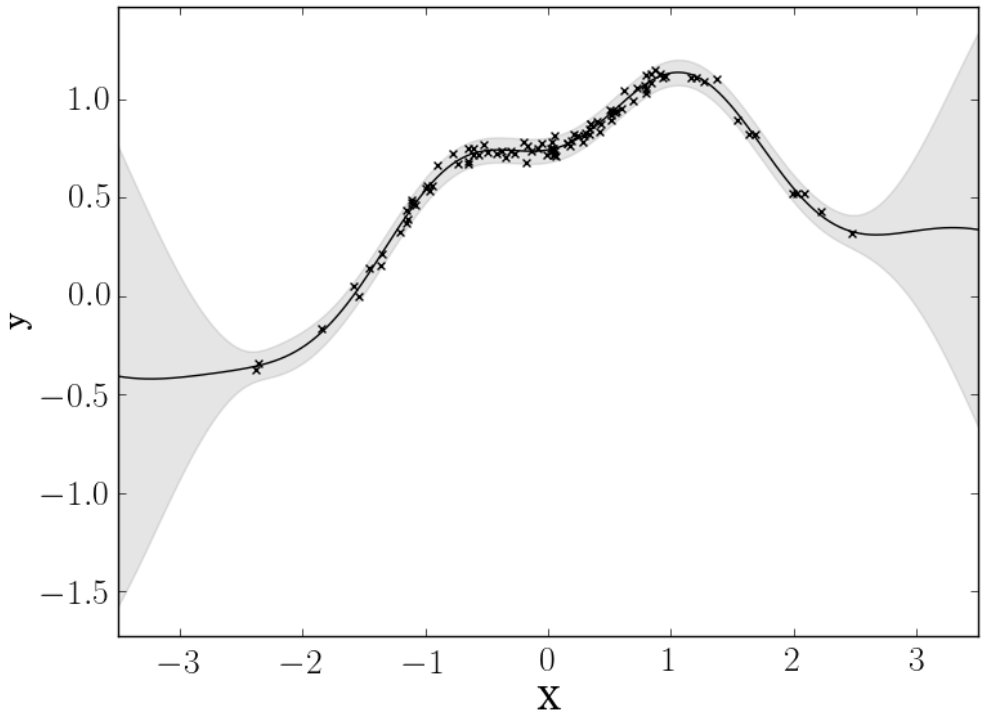


Figure 1.5: A plot of a one-dimensional simulated regression data set, shown with the posterior of a Variational Gaussian Process with 10 inducing points and an exponentiated quadratic kernel. The line displayed is the posterior mean predictive and the 95% credible interval is shaded.

Within the paradigm of stochastic optimisation, the summation over data points in the target function is partially evaluated for b components in the summation, where b is the "batch size". Consequently, an unbiased approximation to the value of the target function is generated and computational costs are reduced from $O(n)$ to $O(b)$ (Hoffman et al., 2013). The method demands a set of global variables, which can be optimised independently of the data. Neither Gaussian processes nor their standard rank-reduced versions possess such a structure, but a further variational approximation enables such inference (Hensman et al., 2013).

The derivation is begun by writing the exact log marginal likelihood with the latent variable $\mathbf{u}$ unmarginalised. The global variables are then provided by a distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$, which is introduced by a factor of $q(\mathbf{u})/q(\mathbf{u})$ within the integral:

$$\log p(\mathbf{y}|\mathbf{X}, \theta, \sigma^2) = \log \int p(\mathbf{y}|\mathbf{u}, \theta) p(\mathbf{u}|\theta) d\mathbf{u} \tag{1.28}$$

$$= \log \int p(\mathbf{y}|\mathbf{u}, \theta) p(\mathbf{u}|\theta) \frac{q(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u}. \tag{1.29}$$

Jensen's inequality can then be used to draw the logarithm within the integral, creating a lower bound:

$$\log \int p(\mathbf{y}|\mathbf{u},\theta)q(\mathbf{u})\frac{p(\mathbf{u}|\theta)}{q(\mathbf{u})}d\mathbf{u} \geq \int q(\mathbf{u}) \log \Big(p(\mathbf{y}|\mathbf{u},\theta)\frac{p(\mathbf{u}|\theta)}{q(\mathbf{u})}\Big)d\mathbf{u} \tag{1.30}$$

$$= \int q(\mathbf{u})\Big(\log p(\mathbf{y}|\mathbf{u},\theta) + \log p(\mathbf{u}|\theta) - \log q(\mathbf{u})\Big)d\mathbf{u}. \tag{1.31}$$

The definition of $\mathcal{L}_1$ in Equation 1.25 can now be used to introduce a further inequality to produce the final lower bound $\mathcal{L}_3$:

$$\log p(\mathbf{y}|\mathbf{X},\theta,\sigma^2) \geq \int q(\mathbf{u})\Big(\log p(\mathbf{y}|\mathbf{u},\theta) + \log p(\mathbf{u}|\theta) - \log q(\mathbf{u})\Big)d\mathbf{u} \tag{1.32}$$

$$\geq \int q(\mathbf{u})\Big(\mathcal{L}_1 + \log p(\mathbf{u}|\theta) - \log q(\mathbf{u})\Big)d\mathbf{u} = \mathcal{L}_3. \tag{1.33}$$

Consequently we have derived $\mathcal{L}_3$, a lower bound to the log posterior expressible as a summation over the data points:

$$\begin{aligned}
\mathcal{L}_3 =& \sum_{i=1}^{n}\Big\{\log \mathcal{N}(y_i|\mathbf{k}_{\mathbf{f}_i\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m},\sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}(\mathbf{S}\Lambda_i)\Big\} - \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u}|\theta)) \\
=& \sum_{i=1}^{n}\Big\{\log \mathcal{N}(y_i|\mathbf{k}_{\mathbf{f}_i\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m},\sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}(\mathbf{S}\Lambda_i)\Big\} \\
&+ \frac{1}{2}\log|\mathbf{S}| - \frac{1}{2}\log|\mathbf{K}_{\mathbf{uu}}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}\right) - \frac{1}{2}\mathbf{m}^T\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m} + \frac{m}{2}, \qquad (1.34) \\
\Lambda_i =& \sigma^{-2}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}i}\mathbf{k}_{i\mathbf{u}}^T\mathbf{K}_{\mathbf{uu}}^{-1}. \qquad (1.35)
\end{aligned}$$

This log posterior can be stochastically optimised with respect to the kernel parameters and the variational parameters $\mathbf{m}$ and $\mathbf{S}$.

We see that this method reduces computational costs to $O(m^3 + bm^2)$, while still targeting the posterior associated with the complete data set. This is a good property, as it allows us to handle arbitrarily large amounts of data with limited computational cost. This is demonstrated in (Hensman et al., 2013), as the authors process data sets with $n = 700,000$ training test points, a large scale in Gaussian Process research.

However, this scalability comes at a cost, as the model optimisation problem has become much more difficult: we have introduced a $m \times m$ matrix, the components of which need to be optimised. Further, since this model includes a second variational approximation, we know that there is a greater distance to the true model, compared to the rank-reduced model. Striking a suitable balance between computational demands and model accuracy is left to the judgement of the practising statistician.

1.3.3 Random Fourier Features

Another widely used method for Gaussian Process scaling is based on the use of Random Fourier Features (Rahimi and Recht, 2007). Bochner's theorem states that the Fourier transform of any stationary positive definite covariance function has strictly positive measure, i.e.:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \int d\omega p(\omega) \exp(i\omega|\mathbf{x}_i - \mathbf{x}_j|), \tag{1.36}$$

where $p(\omega)$ is a normalised probability distribution over frequencies ω. It is then possible to stochastically sample the distribution $p(\omega)$ and hence generate a set of corresponding features $\exp(i\omega|\mathbf{x}_i - \mathbf{x}_j|)$. These can then be used as features in a parametric model, which will be equivalent to an exact GP run with unbiased approximations to the covariance matrix values. However, the computational cost will be cheaper than for a truly nonparametric model, so the model can be run for greater values of n.

The random Fourier features approach has some advantages: it is easy to work using parametric approximations to a nonparametric model. It often performs fairly well in practice, and as we are performing Monte Carlo sampling, there exist asymptotic theoretical results to guarantee convergence to an exact GP. However, there exist some drawbacks: the method only provides discrete parametric approximations to a true GP, there are few non-asymptotic guarantees on how many features need to be sampled for good performance, and the unbiased approximation to the covariance matrix does not correspond to unbiasedness of predictive distributions or the marginal likelihood. Work has been done comparing random Fourier features approximations to the rank-reduced "Nyström" methods, and concluded that the data-independence of the sampled Fourier features leads to weaker generalisation ability (Yang et al., 2012).

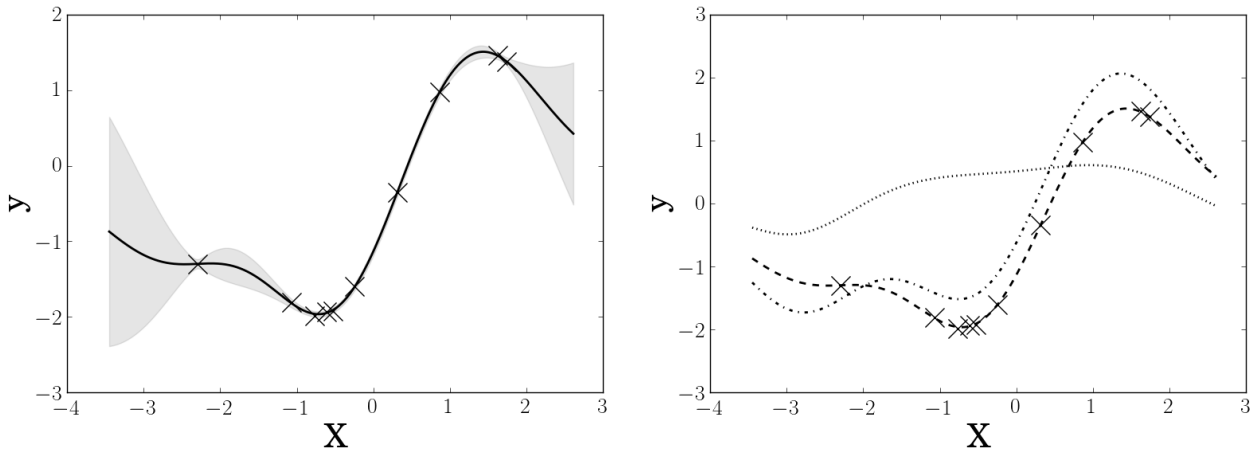
1.3.4 Structured Covariance Matrices

Significant computational gains can be achieved for an exact GP if the covariance matrix has an appropriate structure. For example, work has been pursued that developed a new covariance function which brings about sparsity in the covariance matrix, resulting in cheaper computational costs (Melkumyan and Ramos, 2009).

Of particular relevance to some of the work in this thesis is Kronecker Structure. As will be explained in Chapter 2, if the covariates lie in a complete multidimensional grid and a separable covariance function is used, then the covariance matrix evaluated between all the covariates $\mathbf{X}$ will be expressible as a Kronecker product between dimensions. Consequently, the matrix operations can be separated across dimensions, and the rate-limiting computation for an exact GP becomes the $O(n)$ multiplication with the data vector $\mathbf{y}$.

1.4 Differential Privacy

Another important operational concern in practical Bayesian inference, other than computational cost, is that of privacy preservation. Some data sets are of considerable personal or social sensitivity, and the maintenance of confidentiality of the contributors to the data is of high importance. This complicates the role of the statistician: how is it possible to report statistics derived from data or use insecure computational resources without revealing sensitive details of individuals?



(a) Unperturbed (b) Perturbed

Figure 1.6: A visual demonstration of perturbing a GP to ensure differential privacy. On the left, we see the posterior mean of a GP trained on the plotted data points. On the right, we see a dotted line representing a sample from the prior, a dashed line representing the unperturbed posterior mean, and the dot-dashed line representing the weighted sum of the two: a perturbed, differentially private posterior mean.

One approach to this question is the pursuit of "differential privacy": if none of the statistics or queries provided by the analysis reveal information that would distinguish specific individuals, then we have taken relevant steps to protect confidentiality.

Ideally, we would abide by a differentially private framework, such that an algorithm M returning an answer m would not betray the difference between the true data $\mathcal{D}$, and a data set $\mathcal{D}'$ that differs in one row, i.e. individual queries cannot identify the values of individual data points. Specifically, to be (ϵ, δ)-differentially private, we mean that if an algorithm M, returning an output m, runs on all neighbouring data sets $\mathcal{D}$ and $\mathcal{D}'$, then:

$$P\Big(M(\mathcal{D}) = m\Big) \leq e^{\epsilon} P\Big(M(\mathcal{D}') = m\Big) + \delta \tag{1.37}$$

We see that ϵ limits the amount of privacy forfeited with each query, and the inequality holds with probability $1 - \delta$.

Work has been done exploring how to maintain differential privacy for Gaussian Processes (Smith et al., 2016), building on work looking at functional data (Hall et al., 2013). The Gaussian property of the predictive lends it to the (ϵ, δ)-differentially private mechanisms, as it is possible to maintain conjugacy when adding normally distributed noise.

The proposed method for maintaining differentially private response variables for a Gaussian process involves drawing a sample from the GP prior and adding it to any queried values of the mean, multiplied by a scaling factor $\frac{\Delta c(\delta)}{\epsilon}$, where $c(\delta) \geq \sqrt{2 \log \frac{1.25}{\delta}}$, and Δ is the sensitivity of the algorithm. An example of this mechanism can be seen in Figure 1.6.

The reported predictive variance is left unchanged, as it has no dependence on the response variable.

The methods used ensure differential privacy for the response variable, as individual queries will not reveal enough information to associate values of the response variable with individual data points. The privacy of the covariates is a different consideration: the values of the private covariates are not shared between experts, but no other steps are taken to ensure their privacy. Further work would be more attentive to preserving the privacy of the covariates, although we can claim some strength here by not sharing their values.

Chapter 2

Tucker Variational Gaussian Processes

The use of Tucker decomposition and Kronecker structure of the mean and variance parametrisation of a variational Gaussian Process is explored, enabling the use of very large numbers of inducing points. New variational distributions and lower bounds are introduced, and computational experiments are systematically performed to evaluate their predictive power. Experiments using standard unstructured variational Gaussian Processes are performed, and the results are compared with the Tucker Variational Gaussian Process. This work was a collaboration between Chris Holmes, Michalis Titsias and myself. All of the python code and all of this chapter were written by Owen Thomas.

In this chapter, we pursue the project of using huge numbers of inducing points, by defining them on a grid and using the mathematical formalism of Kronecker structure and Tucker decomposition to provide principled structuring of the variational parametrisation for computational gain. In Section 2.1 we introduce the formalism of Kronecker Structure and Tucker Decomposition and explain the potential computational accelerations. Section 2.2 discusses the use of gridded structure within Gaussian Processes to achieve computational efficiency. In Section 2.3 we introduce a variational Gaussian Process with a Kronecker structured mean vector, and present some preliminary experiments. Section 2.4 introduces the formalism of the full Tucker Variational Gaussian Process. In Section 2.5, a variety of computational experiments are presented and performed, and results are discussed. Section 2.6 compares these results against those taken on a standard unstructured stochastic variational Gaussian Process. Section 2.7 concludes.

2.1 Kronecker Structure and Tucker Decomposition

2.1.1 Kronecker Structure

A Kronecker product $\mathbf{A} \otimes \mathbf{B}$ of an $n_0 \times m_0$ matrix $\mathbf{A}$ and an $n_0' \times m_0'$ matrix $\mathbf{B}$ is defined as the $n_0 n_0' \times m_0 m_0'$ matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \dots & a_{1m_0}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n_0 1}\mathbf{B} & \dots & a_{n_0 m_0}\mathbf{B} \end{pmatrix}. \tag{2.1}$$

We can also extend the operator $\otimes$ to a multiple Kronecker product $\bigotimes_{l=1}^{p}$ over p objects indexed by l, by analogy to the multiple scalar product $\Pi_{l=1}^{p}$. We note, however, that the order of the indices is important for the noncommutative multiple Kronecker product, unlike for the scalar equivalent.

A Kronecker product over p vectors of size $m_0 \times 1$ by the above definition would generate a vector of size $m_0^p \times 1$. However, a reshaping of the resulting vector could generate a matrix of size $m_0^{p/2} \times m_0^{p/2}$, or a tensor object of order p, of size m_0 in each dimension. With a given reshaping scheme, then we can easily establish a one to one correspondence between each of these objects, allowing us to consider them vectors, matrices or tensors, depending on context.

We observe that this notation allows us to represent an object of size $m = m_0^p$ that is exponentially large in dimension p, using components that scale linearly with the number of components p. For this reason, operations on matrices with Kronecker structure can often be much more computationally efficient than unstructured matrices. It is often possible to separate computation across the components in the Kronecker product, effectively reducing the computation from exponential to linear in p.

If we are able to express matrices $\mathbf{A} = \bigotimes_{l=1}^{p} \mathbf{A}^{(l)}$ and $\mathbf{B} = \bigotimes_{l=1}^{p} \mathbf{B}^{(l)}$ as Kronecker products, then the following identities will help us represent common linear algebra operations on $\mathbf{A}$ and $\mathbf{B}$ in a computationally efficient way (Saatci, 2011):

$$\mathbf{AB} = \bigotimes_{l=1}^{p} \mathbf{A}^{(l)} \bigotimes_{l=1}^{p} \mathbf{B}^{(l)} = \bigotimes_{l=1}^{p} \mathbf{A}^{(l)}\mathbf{B}^{(l)}, \tag{2.2}$$

$$\mathbf{A}^{-1} = \left(\bigotimes_{l=1}^{p} \mathbf{A}^{(l)} \right)^{-1} = \bigotimes_{l=1}^{p} (\mathbf{A}^{(l)})^{-1}, \tag{2.3}$$

$$\log |\mathbf{A}| = \log |\bigotimes_{l=1}^{p} \mathbf{A}^{(l)}| = \sum_{l=1}^{p} n_l \log |\mathbf{A}^{(l)}|, \tag{2.4}$$

$$\operatorname{tr}(\mathbf{A}) = \operatorname{tr}(\bigotimes_{l=1}^{p} \mathbf{A}^{(l)}) = \prod_{l=1}^{p} \operatorname{tr}(\mathbf{A}^{(l)}). \tag{2.5}$$

There is also a computationally effective way to evaluate the product of a Kronecker structured matrix of sized $m \times m$ and an unstructured vector of size $m \times 1$ with computational cost $O(pm^{\frac{p+1}{p}}) = O(pm_0^{p+1})$. This algorithm is described in (Saatci, 2011), where it is referred to as `kron_mvprod`. We will use it here: it is reproduced in Appendix A as Algorithm 1.

2.1.2 Tucker Decomposition

A related method is Tucker decomposition, a higher order analogue of singular value decomposition for tensors (Tucker, 1966). The decomposition reduces the number of parameters needed to describe an exponentially large tensor, while imposing less structure than the Kronecker decomposition.

Here we present the Tucker formalism in p dimensions. We wish to represent the pth order tensor $\boldsymbol{\alpha}$ of size $m = m_0^p$ via a Tucker decomposition: we notate this by $\boldsymbol{\alpha} = U^{(l)} \times_{l=1}^{p} W$, by which we mean:

$$\boldsymbol{\alpha}_{i_1,i_2,\ldots,i_p} = \sum_{j_1} \sum_{j_2} \cdots \sum_{j_p} U^{(1)}_{i_1 j_1} U^{(2)}_{i_2 j_2} \ldots U^{(p)}_{i_p j_p} W_{j_1,j_2,\ldots,j_p}, \tag{2.6}$$

where W is the core tensor of size q_0^p, with $q_0 < m_0$, and the U matrices are factor matrices of size $m_0 \times q_0$. The core tensor W still scales exponentially in size with the dimension. However, we are free to choose a q_0 much smaller than m_0, heavily reducing the computational load. Taking this to an extreme, if we choose $q_0 = 1$, we remove all exponential scaling and recover the Kronecker product to within a scalar multiplicative factor.

There exist similar identities to the Kronecker product to evaluate common linear algebra operations with a Tucker-structured object. Most relevantly, it is possible to take products of Tucker-decomposed objects and Kronecker-structured objects without evaluating anything of size m. Specifically, if there is a matrix, $\mathbf{A} = \bigotimes_{l=1}^{p} \mathbf{A}^{(l)}$, then we can express the product $\mathbf{A}$ with a vectorised $\boldsymbol{\alpha}$ by the following:

$$\mathbf{A}\boldsymbol{\alpha} = \left(\bigotimes_{l=1}^{p} \mathbf{A}^{(l)} \right) \left(U^{(l)} \times_{l=1}^{p} W \right) = \left(\mathbf{A}^{(l)} U^{(l)} \right) \times_{l=1}^{p} W.$$

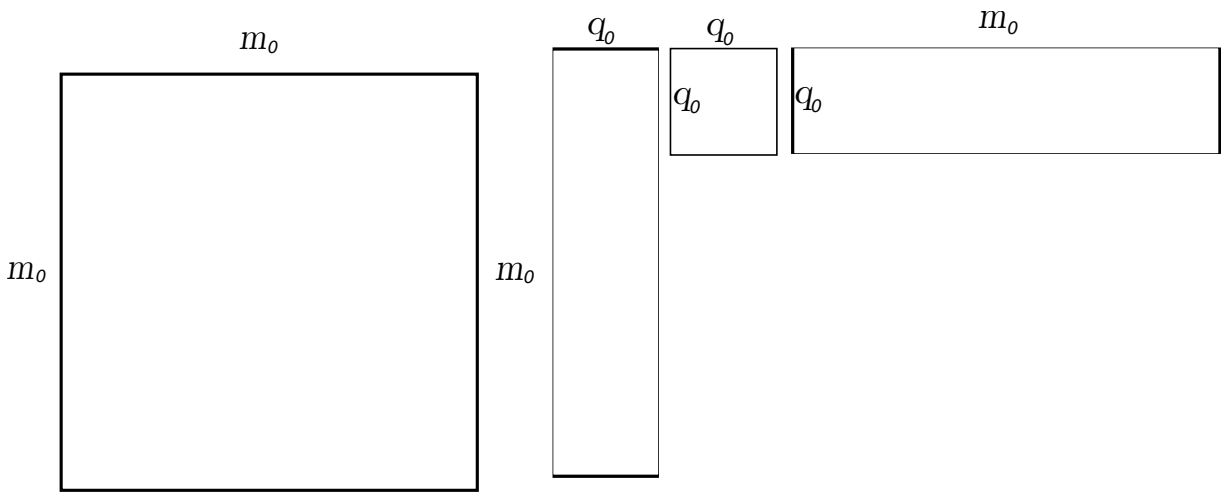
We see that we have taken the product of an $n \times m$ object and an $m \times 1$ vector by only performing a series of matrix multiplications between each of the components of the Kronecker product and the factor matrices of the Tucker decomposition.

There are strong analogies to a truncated Singular Value Decomposition (SVD) of a second order matrix. A Tucker decomposition can be considered a similar operation performed on a higher order tensor object: the relationship can be seen in Figure 2.1 and Figure 2.2.

Similarly, we can take the inner product of two Tucker-decomposed vectors efficiently:

$$\boldsymbol{\alpha}^T \boldsymbol{\alpha} = \left(U^{(l)T} \times_{l=1}^{p} W^T \right) \left(U^{(l)} \times_{l=1}^{p} W \right) = W^T \left(U^{(l)T} U^{(l)} \times_{l=1}^{p} W \right).$$

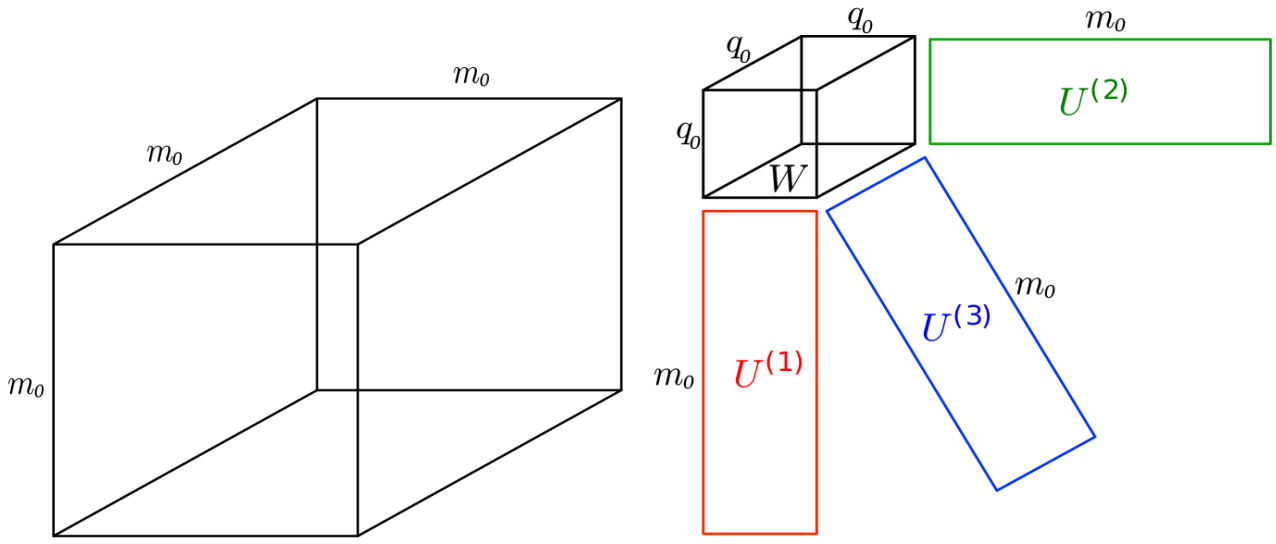
The matrix products $U^{(l)T} U^{(l)}$ will generate p matrices of size $q_0 \times q_0$ with computational cost $O(p m_0 q_0^2)$, avoiding evaluating any objects of size $O(m)$. We can then use the efficient Algorithm 1 to evaluate the object $U^{(l)T} U^{(l)} \times_{l=1}^{p} W$, and then perform an inner product between two objects of size q_0^p. Consequently, we have performed an inner product between two objects of size $m = m_0^p$ with computational cost $O(p q_0^{p+1}) + O(p m_0 q_0^2)$, which for $q_0 < m_0$ can represent a substantial improvement.



(a) A full square matrix of size m_0 by m_0.

(b) A singular value decomposition (SVD) of a matrix of size m_0 by m_0 with truncation of the core matrix to size q_0 by q_0.

Figure 2.1: A representation of a truncated SVD of a matrix of size m_0 by m_0.



(a) A full tensor of size m_0 by m_0 by m_0.

(b) A Tucker decomposition (SVD) of a tensor of size m_0 by m_0 by m_0 with truncation of the core tensor to size q_0 by q_0 by q_0.

Figure 2.2: A representation of a Tucker decomposition of a tensor of size m_0 by m_0 by m_0. The diagram was taken with permission and some editing from (Kim et al., 2016).

2.2 Gridded Structure in Gaussian Processes

2.2.1 Gridded Exact Gaussian Processes

Within the context of Gaussian Processes, work has been done using Kronecker structure to accelerate exact Gaussian Processes. Two criteria need to be fulfilled. Firstly, the data need to occupy a full rectangular grid of size n: such data often appear in image analysis and geospatial statistics. Secondly, a covariance function must be used that is separable across dimensions, i.e. $k(X_i, X_j) = \Pi_{l=1}^{p} k^{(l)}(X_i^{(l)}, X_j^{(l)})$, a reasonably common property among widely used kernel functions. If these conditions are fulfilled, the covariance matrix of the latent function evaluated at all of the data takes on a Kronecker product structure, such that it can be expressed as a multiple Kronecker product between the covariance matrices associated with each dimension, i.e.:

$$\mathbf{K} = \bigotimes_{l=1}^{p} \mathbf{K}^{(l)}. \tag{2.7}$$

We can see that the matrix operations in the marginal likelihood of an exact Gaussian Process can be performed using matrix operations decomposed across the Kronecker product, specifically the computation of the log determinant and the matrix inverses. The $O(pn^{\frac{p+1}{p}})$ Algorithm 1 can be used to efficiently compute a product between a Kronecker-structured matrix and a vector of length n. This is used in the marginal likelihood to calculate the product between the matrix inverse $\mathbf{K}^{-1}$ and the data vector $\mathbf{y}$.

We observe that the gridded structure of the data suggests that the computational cost $O(pn^{\frac{p+1}{p}})$ necessarily scales exponentially with dimension. (Saatci, 2011) demonstrates the power of Kronecker methods to evaluate marginal likelihoods, but struggles with the constraints of severe exponential $O(pn^{\frac{p+1}{p}}) + O(pn^{\frac{3}{p}})$ scaling with dimension, restriction of the covariates to a grid and the use of a separable covariance function.

2.2.2 Gridded Approximate Gaussian Processes

It is tempting to extend the use of Kronecker structure to the inducing point framework to allow for the computationally cheap use of many gridded inducing points. However, the computational gains do not emerge automatically with standard inducing point methods.

If we simply take a variational Gaussian Process and impose Kronecker structure on the inducing points, we encounter the following issue. Ordinarily, we would be compelled to perform a matrix decomposition of:

$$\mathbf{K_{uu}} + \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathbf{K_{uf}}_i \mathbf{K}_{\mathbf{f}_i \mathbf{u}}.$$

If we now introduce Kronecker structure into the inducing points, then the object of interest becomes:

$$\bigotimes_{l=1}^{p} \mathbf{K}_{\mathbf{uu}}^{(l)} + \frac{1}{\sigma^2} \sum_{i=1}^{n} \bigotimes_{l=1}^{p} \mathbf{K}_{\mathbf{uf}_i}^{(l)} \mathbf{K}_{\mathbf{f}_i\mathbf{u}}^{(l)}.$$

However, the introduction of Kronecker structure does not help the decomposition of the entire object: here we encounter the fact that sums of products are not equivalent to products of sums.

However, it is possible to usefully introduce Kronecker structure to approximate Gaussian Processes for computational advantage. In a variational context, work has been done that uses Kronecker computational methods with the explicitly parametrised variational bound.

"Blitzkriging" imposes Kronecker structure on the variational covariance parametrisation (Nickson et al., 2015). This uses subsampling of data for the inference, removing the dependence of the computational scaling on n, and the dependence on the number of inducing points is reduced to $O(m)$. The model exhibits gains in performance compared to the state of the art, but still contains implicit exponential scaling in dimension with fully gridded inducing points.

KISS-GP also has similar ambitions to use Kronecker structured inducing points, combining them with local interpolation to reduce computational demands. However, it similarly exhibits $O(m)$ scaling that is implicitly exponential in the number of dimensions (Wilson and Nickisch, 2015).

2.3 Kronecker Structured Variational Mean Vector

Here we explore imposing Kronecker structure on the variational mean vector, reducing the computation cost to linear in the number of dimensions, ideally without unduly compromising the model's expressive capability. We also pursue new preconditioning and representations for the variational distribution parameters.

There are several ways of interpreting a product structure across the variational mean vector. It should be stressed that this is not the same thing as performing a product of Gaussian Processes across dimensions: the product of several GPs would not itself be a GP, whereas this approximation is.

There are several existing statistical models which form a predictive distribution by taking a product across dimensions. Log-linear models have a latent variable formed from a sum over covariates, which is then combined with a logarithmic likelihood to form a posterior predictive which is effectively a product across features (McCullagh, 1984). Similarly, the hazard function for a Cox model in survival analysis features an exponentiated linear sum, bringing about a product-like likelihood (Cox, 1992). It is worth noticing that each of these models have non-negative response variables, whereas we are dealing with real-valued responses. This may have the effect of destabilising the inference for the product GP model.

We introduce preconditioning to the variational mean, and a representation of the variational covariance inspired by (Opper and Archambeau, 2009). The new representation reduces correlations between variational parameters, enabling more effective inference. The methods

are compatible with introducing Kronecker structure, reducing the cost to $O(pm_0^3)$. We suggest the following variational distribution:

$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}\middle|\mathbf{K_{uu}}\boldsymbol{\alpha}, \mathbf{K_{uu}} - \mathbf{K_{uu}}(\mathbf{K_{uu}} + \Sigma)^{-1}\mathbf{K_{uu}}\right), \tag{2.8}$$

where the variational parameters are $\boldsymbol{\alpha}$, a vector of length m, and Σ, a diagonal matrix, the m diagonal components of which are positive. The introduction of preconditioning to the mean representation should help decorrelate the elements of the variational mean vector and also remove one of the matrix inversion operations in the variational bound. Similarly, the introduction of the diagonal matrix Σ should help decorrelate the components of the variational covariance representation, as the matrix Σ will be standing in for the diagonal elements of the matrix $\mathbf{K_{uf}}\mathbf{K_{fu}}/\sigma^2$. Additionally, reducing the number of variational parameters to $2m$ should significantly simplify the optimisation process.

Recalling the bound introduced for a standard stochastic variational Gaussian Process in Equation 1.35:

$$\begin{aligned} F =& \sum_{i=1}^{n}\left\{\log\mathcal{N}(y_i|\mathbf{k_{f_iu}}\mathbf{K_{uu}^{-1}}\mathbf{m}, \sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}(\mathbf{S}\Lambda_i)\right\} \\ &+ \frac{1}{2}\log|\mathbf{S}| - \frac{1}{2}\log|\mathbf{K_{uu}}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K_{uu}^{-1}}\mathbf{S}\right) - \frac{1}{2}\mathbf{m}^T\mathbf{K_{uu}^{-1}}\mathbf{m} + \frac{m}{2}, \qquad (2.9) \\ \Lambda_i =& \sigma^{-2}\mathbf{K_{uu}^{-1}}\mathbf{k_{ui}}\mathbf{k_{iu}^T}\mathbf{K_{uu}^{-1}}, \qquad (2.10) \end{aligned}$$

we now introduce the new parametrisation of $\mathbf{m} = \mathbf{K_{uu}}\boldsymbol{\alpha}$ and $\mathbf{S} = \mathbf{K_{uu}} - \mathbf{K_{uu}}(\mathbf{K_{uu}} + \Sigma)^{-1}\mathbf{K_{uu}}$:

$$\begin{aligned} F =& \sum_{i=1}^{n}\left\{\log\mathcal{N}(y_i|\mathbf{k_{f_iu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uu}}\boldsymbol{\alpha}, \sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}\left((\mathbf{K_{uu}} - \mathbf{K_{uu}}(\mathbf{K_{uu}} + \Sigma)^{-1}\mathbf{K_{uu}})\Lambda_i\right)\right\} \\ &+ \frac{1}{2}\log|\mathbf{K_{uu}} - \mathbf{K_{uu}}(\mathbf{K_{uu}} + \Sigma)^{-1}\mathbf{K_{uu}}| - \frac{1}{2}\log|\mathbf{K_{uu}}| \\ &- \frac{1}{2}\mathrm{tr}\left(\mathbf{K_{uu}^{-1}}(\mathbf{K_{uu}} - \mathbf{K_{uu}}(\mathbf{K_{uu}} + \Sigma)^{-1}\mathbf{K_{uu}})\right) - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K_{uu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uu}}\boldsymbol{\alpha} + \frac{m}{2}. \qquad (2.11) \end{aligned}$$

After some algebra and cancellation, F becomes:

$$\begin{aligned} F =& \sum_{i=1}^{n}\log\mathcal{N}\left(y_i|\mathbf{k_{f_iu}}\boldsymbol{\alpha}, \sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\mathbf{k_{f_iu}}\mathbf{K_{uu}^{-1}}\mathbf{k_{uf_i}} - \mathbf{k_{f_iu}}\left(\mathbf{K_{uu}} + \Sigma\right)^{-1}\mathbf{k_{uf_i}}\right) \\ &- \frac{1}{2\sigma^2}\sum_{i=1}^{n}\mathbf{k_{f_if_i}} + \frac{1}{2\sigma^2}\sum_{i=1}^{n}\mathbf{k_{f_iu}}\mathbf{K_{uu}^{-1}}\mathbf{k_{uf_i}} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K_{uu}}\boldsymbol{\alpha} \\ &+ \frac{1}{2}\mathrm{tr}\left(\left(\mathbf{K_{uu}} + \Sigma\right)^{-1}\mathbf{K_{uu}}\right) + \frac{1}{2}\log|\Sigma| - \frac{1}{2}\log|\mathbf{K_{uu}} + \Sigma|. \qquad (2.12) \end{aligned}$$

It is then possible to introduce gridded structure on the inducing points, bringing about Kronecker structure of the matrices of covariance function evaluations, i.e.:

$$\mathbf{K_{uu}} = \bigotimes_{l=1}^{p} \mathbf{K}_{\mathbf{uu}}^{(l)}, \quad \mathbf{k}_{\mathbf{f}_i\mathbf{u}} = \bigotimes_{l=1}^{p} \mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}. \tag{2.13}$$

Further, we can introduce Kronecker structure into the variational parametrisation, resulting in a new variational distribution:

$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\bigotimes_{l=1}^{p} \mathbf{K}_{\mathbf{uu}}^{(l)}\boldsymbol{\alpha}^{(l)}, \bigotimes_{l=1}^{p}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} - \mathbf{K}_{\mathbf{uu}}^{(l)}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1}\mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right). \tag{2.14}$$

The Kronecker products within the scalar quantities in the bound resolve into scalar products, bringing about the final representation of the bound:

$$\begin{aligned}
F = &\sum_{i=1}^{n} \log \mathcal{N}\left(y_i|\prod_{l=1}^{p}\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}\boldsymbol{\alpha}^{(l)}, \sigma^2\right) \\
&- \frac{1}{2\sigma^2}\sum_{i=1}^{n}\prod_{l=1}^{p}\left(\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}(\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)} - \mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)}\right) \\
&- \frac{1}{2\sigma^2}\sum_{i=1}^{n}\mathbf{k}_{\mathbf{f}_i\mathbf{f}_i} + \frac{1}{2\sigma^2}\sum_{i=1}^{n}\prod_{l=1}^{p}\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}(\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)} \\
&- \frac{1}{2}\prod_{l=1}^{p}\left((\boldsymbol{\alpha}^{(l)})^T\mathbf{K}_{\mathbf{uu}}^{(l)}\boldsymbol{\alpha}^{(l)}\right) - \frac{1}{2}\prod_{l=1}^{p}\left(m_0 - \mathrm{tr}\left(\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1}\mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right) \\
&+ \frac{1}{2}\sum_{l=1}^{p} m_0^{p-1}\log|\Sigma^{(l)}| - \frac{1}{2}\sum_{l=1}^{p} m_0^{p-1}\log|\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}| + \frac{m_0^p}{2}.
\end{aligned} \tag{2.15}$$

We observe that every component in the bound is composed of products of separate computations across the dimensions, such that it can be evaluated with complexity $O(pm_0^3)$. Consequently, we can run a model with a number of inducing points that is exponential in the number of dimensions, but with computational demands that are linear in p and n.

2.3.1 Sum of Kronecker Products

Initial experiments indicated that the above bound possesses limited expressive capability, as it can only adequately capture data whose generating mean function forms a product across dimensions. Consequently, we attempt to increase the generality of the bound by representing the mean $\mathbf{m}$ and covariance $\mathbf{S}$ as a sum over K Kronecker products:

$$\begin{aligned}
\mathbf{m} =& \frac{1}{K}\sum_{k=1}^{K}\bigotimes_{l=1}^{p}\mathbf{K}_{\mathbf{uu}}^{(l)}{\boldsymbol{\alpha}_k}^{(l)}, \\
\mathbf{S} =& \frac{1}{K}\sum_{k=1}^{K}\bigotimes_{l=1}^{p}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} - \mathbf{K}_{\mathbf{uu}}^{(l)}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma_k^{(l)}\right)^{-1}\mathbf{K}_{\mathbf{uu}}^{(l)}\right).
\end{aligned}$$

This maintains the appealing $O(pm_0^3)$ scalability with a new linear scaling in K, while increasing the flexibility of the variational distribution. Mathematically, this results in a new variational distribution $q(\mathbf{u})$ and variational bound F:

$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\frac{1}{K}\sum_{k=1}^{K}\bigotimes_{l=1}^{p}\mathbf{K}_{\mathbf{uu}}^{(l)}{\boldsymbol{\alpha}_k}^{(l)}, \frac{1}{K}\sum_{k=1}^{K}\bigotimes_{l=1}^{p}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} - \mathbf{K}_{\mathbf{uu}}^{(l)}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma_k^{(l)}\right)^{-1}\mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right), \tag{2.16}$$

$$\begin{aligned}
F = &\sum_{i=1}^{n}\log\mathcal{N}\left(y_i|\frac{1}{K}\sum_{k=1}^{K}\prod_{l=1}^{p}({\boldsymbol{\alpha}_k}^{(l)})^T\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}, \sigma^2\right) \\
&- \frac{1}{2\sigma^2 K}\sum_{k=1}^{K}\sum_{i=1}^{n}\prod_{l=1}^{p}\left(\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}(\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)} - \mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma_k^{(l)}\right)^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)}\right) \\
&- \frac{1}{2\sigma^2}\sum_{i=1}^{n}\mathbf{k}_{\mathbf{f}_i\mathbf{f}_i} + \frac{1}{2\sigma^2}\sum_{i=1}^{n}\prod_{l=1}^{p}\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)}(\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1}\mathbf{k}_{\mathbf{uf}_i}^{(l)} \\
&- \frac{1}{2K}\sum_{k=1}^{K}\prod_{l=1}^{p}\left(({\boldsymbol{\alpha}_k}^{(l)})^T\mathbf{K}_{\mathbf{uu}}^{(l)}{\boldsymbol{\alpha}_k}^{(l)}\right) - \frac{1}{2K}\sum_{k=1}^{K}\prod_{l=1}^{p}\left(m_0 - \mathrm{tr}\left(\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma_k^{(l)}\right)^{-1}\mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right) \\
&+ \frac{1}{2K}\sum_{k=1}^{K}\sum_{l=1}^{p}m_0^{p-1}\log|\Sigma_k^{(l)}| - \frac{1}{2}\sum_{l=1}^{p}m_0^{p-1}\log|\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma_k^{(l)}| + \frac{m_0^p}{2}.
\end{aligned} \tag{2.17}$$

The new bound should exhibit more generality than the simple Kronecker bound: there is no individual product across dimensions in this model, so it should be possible to increase the size K to increase the expressive capacity of the model. We also notice that the computations in the sum over k are separable, which allows us to pursue a variety of different learning algorithms.

We explored the usage of learning multiple terms in the sum simultaneously, or pursue a greedy algorithm to learn each new term in the variational sum on the residuals of the previous models. It has not been clear which method is most efficient: there are good computational reasons to greedily separate inference between components in the sum, but it will remove the ability of the components to exchange information during learning, making inference more challenging.

2.3.2 Preliminary Results

We present in Table 2.1 some initial results demonstrating the performance of the model on the Boston housing data set. The covariates and response variable were standardised to have zero mean and unit variance. We present the training and test RMSE on the unit variance test response variable as we successively add components to the sum of Kroneckers variational bound:

Number of components K	Train RMSE	Test RMSE
1	0.6722	0.8878
2	0.6790	1.0002
3	0.6897	1.1766
4	0.7050	1.3663
5	0.7232	1.5578

Table 2.1: The training and test RMSEs of the sum of Kroneckers variational bound on the Boston data set, run with increasing values of K.

The results are not competitive with other methods: it is not clear whether the bound is too heavily structured, or whether the inference is too challenging for the methods being employed. The heavy imbalance between training and test error suggests that overfitting is a problem. It is also alarming that introducing more components to the sum produces a uniformly negative effect on the RMSE performance. Consequently, alternative methods were pursued to exploit product structure in a principled and inferentially tractable fashion.

2.4 Tucker Structured Variational Mean Vector

Some recent work has explored the use of Tucker decomposition within the "weight space" view of GPs, introducing a new model called the "TuckerGP" (Kim et al., 2016). For the TuckerGP, Tucker decomposition was used to efficiently represent the vector of weights in a random Fourier features GP approximation: for our work, it can be used to represent the preconditioned mean vector in a principled and efficient fashion.

Specifically, we redefine the preconditioned mean vector as $\boldsymbol{\alpha} \sim U^{(l)} \times_{l=1}^{p} W$, where each $U^{(l)}$ is a factor matrix of size $(m_0 \times q_0)$ and W is a vector of length ${q_0}^p$. We keep the variational covariance parametrised with a Kronecker structure as before, giving the following variational distribution:

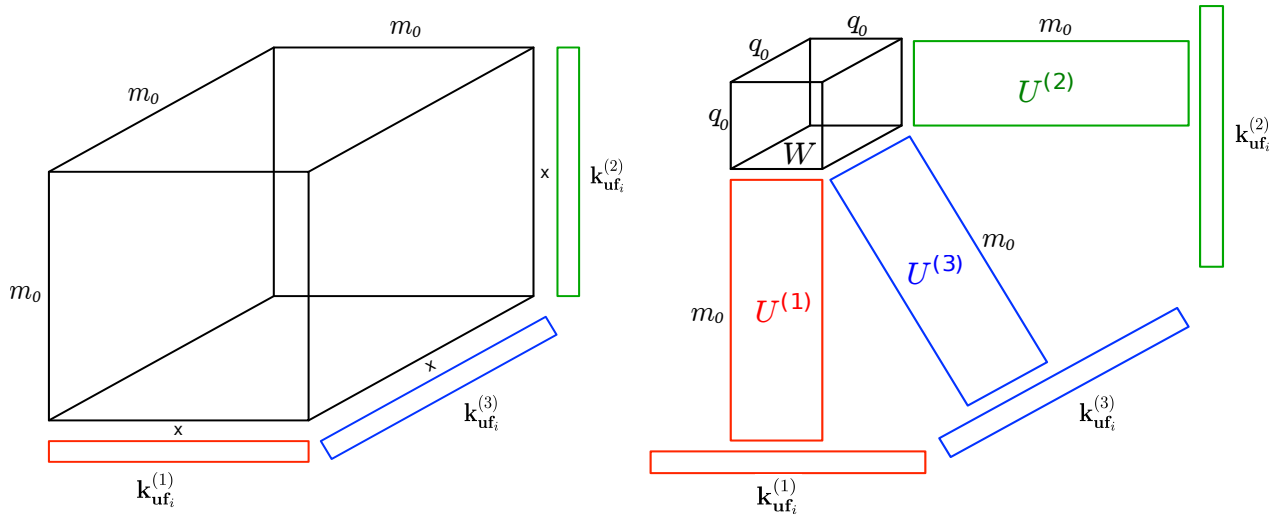
$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u} \middle| \mathbf{K}_{\mathbf{uu}}^{(l)} U^{(l)} \times_{l=1}^{p} W, \bigotimes_{l=1}^{p} \left(\mathbf{K}_{\mathbf{uu}}^{(l)} - \mathbf{K}_{\mathbf{uu}}^{(l)} \left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1} \mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right). \tag{2.18}$$

2.4.1 Variational Bound

If we use the above variational parametrisation for the variational bound, and again assume gridded structure on the inducing points, we derive the following expression:

$$\begin{aligned}
F = &\sum_{i=1}^{n} \log \mathcal{N}\left(y_i | W^T \times_{l=1}^{p} U^{(l)T} \mathbf{k}_{\mathbf{uf}_i}^{(l)}, \sigma^2\right) \\
&- \frac{1}{2\sigma^2} \sum_{i=1}^{n} \prod_{l=1}^{p} \left(\mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)} (\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1} \mathbf{k}_{\mathbf{uf}_i}^{(l)} - \mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)} \left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1} \mathbf{k}_{\mathbf{uf}_i}^{(l)}\right) \\
&- \frac{1}{2\sigma^2} \sum_{i=1}^{n} \mathbf{k}_{\mathbf{f}_i\mathbf{f}_i} + \frac{1}{2\sigma^2} \sum_{i=1}^{n} \prod_{l=1}^{p} \mathbf{k}_{\mathbf{f}_i\mathbf{u}}^{(l)} (\mathbf{K}_{\mathbf{uu}}^{(l)})^{-1} \mathbf{k}_{\mathbf{uf}_i}^{(l)} \\
&- \frac{1}{2} W^T \times_{l=1}^{p} \left(U^{(l)T} \mathbf{K}_{\mathbf{uu}}^{(l)} U^{(l)}\right) \times_{l=1}^{p} W - \frac{1}{2} \prod_{l=1}^{p} \left(m_0 - \operatorname{tr}\left(\left(\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}\right)^{-1} \mathbf{K}_{\mathbf{uu}}^{(l)}\right)\right) \\
&+ \frac{1}{2} \sum_{l=1}^{p} m_0^{p-1} \log |\Sigma^{(l)}| - \frac{1}{2} \sum_{l=1}^{p} m_0^{p-1} \log |\mathbf{K}_{\mathbf{uu}}^{(l)} + \Sigma^{(l)}| + \frac{m_0^p}{2}.
\end{aligned} \tag{2.19}$$

Again, we see that much of the computation has been separated into products across p, with the exception of the products with the vector W. Derivatives of this expression were taken, enabling the use of stochastic gradient descent for the joint learning of the variational parameters and the hyperparameters. We call the resulting model the Tucker Variational



(a) An unstructured variational mean tensor

(b) A structured variational mean tensor

Figure 2.3: A representation of the variational mean in three dimensions evaluated at data point i, in a completely unstructured form on the left and in a Tucker-structured form on the right. We see that the product between the variational bound and the covariances are separated into products between each $U^{(l)}$ and the corresponding $\mathbf{k}^{(l)}_{\mathbf{uf}_i}$. The diagram was taken with permission and some editing from (Kim et al., 2016).

Gaussian Process (TVGP). It is worth observing that with $q_0 = 1$ we recover the Kronecker bound earlier with a multiplicative constant: the Tucker bound is a principled generalisation of the previous model.

We expect computational scaling of the form $O(pq_0^{p+1}) + O(pm_0q_0^2) + O(pm_0^3)$, with each term being respectively introduced by the use of Algorithm 1, the p products $U^{(l)T}U^{(l)}$ and the p matrix decompositions of the $\mathbf{K}^{(l)}_{\mathbf{uu}}$.

As an example, we notice in the variational bound that we need to calculate the quantity $\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}$. Given gridded inducing points, we have Kronecker structure on the covariance matrix $\mathbf{K}_{\mathbf{uu}}$, i.e.:

$$\mathbf{K}_{\mathbf{uu},\{h_1h_2\ldots h_p,i_1i_2\ldots i_p\}} = \prod_{l=1}^{p} \mathbf{K}^{(l)}_{\mathbf{uu},\{h_l,i_l\}}. \tag{2.20}$$

We see that it is possible to evaluate the product $\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}$ efficiently using the Tucker decomposition. To evaluate the element $[\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}]_{h_1h_2\ldots h_p}$, we have:

$$\begin{aligned}
[\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}]_{h_1h_2\ldots h_p} &= \sum_{i_1}\sum_{i_2}\cdots\sum_{i_p} \mathbf{K}_{\mathbf{uu},\{h_1h_2\ldots h_p,i_1i_2\ldots i_p\}}\boldsymbol{\alpha}_{i_1i_2\ldots i_p} \\
&= \sum_{j_1}\sum_{j_2}\cdots\sum_{j_p}\prod_{l=1}^{p}\Big(\sum_{i_l}\mathbf{K}^{(l)}_{\mathbf{uu},\{h_l,i_l\}}U^{(l)}_{i_lj_l}\Big)W_{j_1j_2\ldots j_p}.
\end{aligned} \tag{2.21}$$

We see that we have divided the computation of the product between the covariances and the mean vector into separate sums for each dimension over each i_l. A visual representation can be seen in Figure 2.3. The multiple summation over the indices j_l of the core tensor is exponential in dimension, but is now independent of the number of inducing points, so we

will not have to compromise coverage of covariate space for computational gains as we did previously.

2.5 Computational Experiments

2.5.1 Experimental Design

2.5.1.1 Tucker Variational GP Experimental Design

There are several properties of the experiments that we are interested in varying, as they may all affect the properties of the inference for the TVGP. Here we present the parameters that were varied: experiments were performed exhaustively over every combination of these values.

Firstly, we use three different data sets to see how the model responds to different challenges. Specifically, we sought data sets with a large number of data points, and varying dimensionalities. The first data set is the "Jutland" data set, with $n = 434,874$ data instances, with which we predict the altitude of a given point given the $p = 2$ dimensions of longitude and latitude. The second data set is the Combined Cycle Power Plant ("CCPP") data set, by which we predict power plant energy output with $p = 4$ covariates and $n = 9,568$ data instances. The third data set is the household power consumption data set ("Household"), with which household power consumption is predicted using $p = 8$ covariates and $n = 2,075,259$ data instances. The data sets were partitioned such that 90% of the data were used for training and 10% for testing predictions, and normalised such that the covariates and response variable each have zero mean and unit variance. Consequently, test RMSEs will be reported as a consistent measure of explained variance.

Secondly, we can vary the value of q_0, which will determine the size of the core vector W. q_0 takes the values 1, 2 and 3. We notice that with $q_0 = 1$, we recover the Kronecker regime as explored earlier.

Thirdly, we vary the value of m_0, which determines the number of inducing points ${m_0}^p$ in the variational approximation. m_0 takes on the values of 3, 5, 10 and 20 in these experiments. Excitingly, for $m_0 = 20$ and the 8 dimensional Household data set, we are implicitly using ${m_0}^p = 25.6$ billion inducing points, which to our knowledge is an unprecedented amount.

Fourthly, we can vary the batch size b. We will be performing Stochastic Variational Inference by taking batches of data of size b at each stage of optimisation. This will introduce some noise into the evaluations of the gradients, with a smaller batch size introducing greater variance to the gradient realisations and potentially making inference more challenging.

Finally, we can vary the random seed of the random number generator used with all integer values from 0 to 9. This should ensure that there are sufficient quantities of results to allow for bulk comparison between different experimental parameter regimes.

Some properties of the optimisation were held constant between the runs: the stochastic optimisation algorithm used was Adagrad, run for $50,000$ iterations. The GP covariance uses an ARD exponentiated quadratic kernel. The kernel power and noise variance were held constant for all of the inference at values of 1.0 and 0.1, as allowing them to vary made the optimisation

much less stable. The simulations were run on a desktop computer, with 8 cores and 7.69 GB of memory. The simulations were coded in Python, using the standard NumPy libraries for the numerics.

2.5.2 GPy SVIGP

We also present results taken using a standard SVIGP from the GPy package, following the methodology in (Hensman et al., 2013). It uses a more general variational distribution, with no structuring of the mean vector, and an $O(m^2)$ parametrisation of the variational covariance. The GP covariance uses an ARD exponentiated quadratic kernel.

The optimisation was performed "out of the box" with GPy's default optimisation of 25,000 iterations of the AdaDelta algorithm. The computation was performed on the same computational system as the TVGP inference, such that we can directly compare computational times between methods.

Properties of the SVIGP results will be presented alongside the TVGP results, with means and medians taken over all the experimental conditions considered. Further discussion of the SVIGP results and comparison to the TVGP will occur when all results have been presented.

2.5.3 SVIGP Experimental Design

There were four properties of the experiments that were varied between runs for the SVIGP:

Firstly, the number of inducing points m took values of 10, 50, 200 and 500. These values will represent a range of expressiveness of variational distributions, but also will explore the anticipated $O(m^3)$ computational scaling.

Secondly, the batch size b was varied, taking values of 1, 10, 50 and 100. This should demonstrate how robust the inference procedure is to various batch sizes, while also demonstrating the expected $O(b)$ scaling.

Thirdly, the data set was varied, using the same Jutland, CCPP and Household Power data sets used for the TVGP experiments.

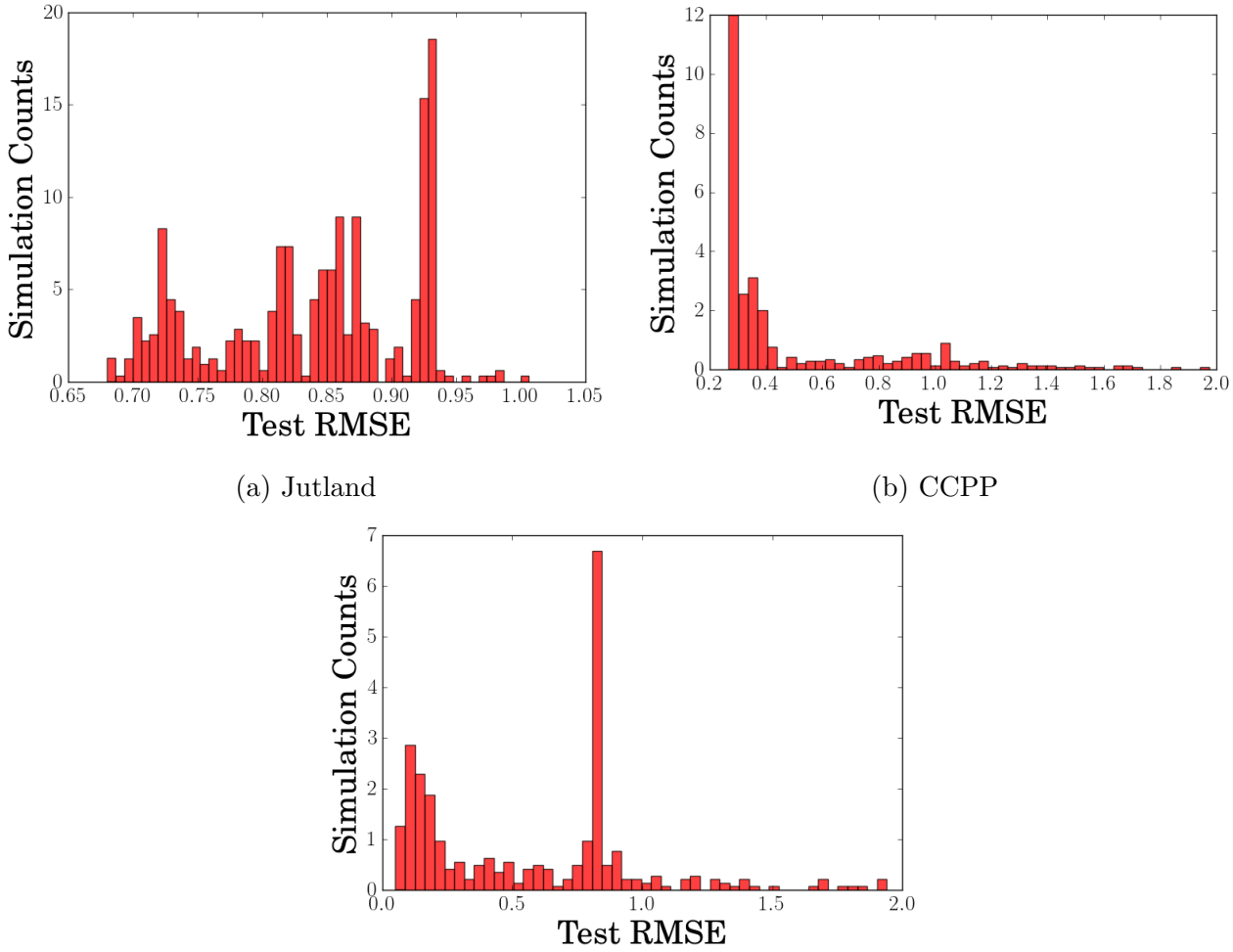
Finally, the random seed was initiated taking all the integer values from 0 to 9, ensuring there are plenty of independent results for analysis.

2.5.4 Tucker Variational GP Results

2.5.4.1 Histogram of Test RMSEs

Here we present histograms of all of the Tucker Variational GP runs performed, separated by data set. This will demonstrate the distribution of results across experimental regimes, and give some insight into the challenges involved in the inference.

For the Jutland data set, we see in Figure 2.4 that there is a distribution of test RMSEs between approximately 0.68 and 1.01. There are no spurious runs with values of test RMSE much greater than 1, although we do observe a spike of poorly optimised runs at approximately 0.93. We do, however, observe a significant population of runs between 0.68 and 0.75.



(a) Jutland

(b) CCPP

(c) Household

Figure 2.4: Histograms of all test RMSEs from the TVGP run on the Jutland, CCPP and Household data sets. The CCPP and Household test RMSEs have been truncated to have values less than 2.

For the CCPP data set in Figure 2.4, we see that the test RMSE is much more closely grouped around the minimum values of approximately 0.25, with a long tail towards large values of the test RMSE. The histogram is truncated with test RMSE < 2, as some of the runs occasionally returned extremely high values of the test RMSE, with a worst-performing run of 44.60.

Finally, we see the histogram for the test RMSE of all the runs on the Household data set, similarly truncated to have test RMSE < 2. The distribution is broadly bimodal, with one set of runs clustered near the minimum values, and one set of poorly optimised runs at approximately 0.8. The results also exhibited a very long tail to very large values of the test RMSE, with the maximum value a surprising $9,579.58$.

Given the heavy skew and extreme outliers observed in some of the test RMSE results, we will report the median with interquartile ranges along with the mean and standard deviation of the test RMSEs from groups of runs. It is possible that the products taken across larger number of dimensions can be very unstable, resulting in the few very large test RMSEs.

2.5.4.2 Dependence of Results on q_0

Here we represent the bulk statistics of the experimental results stratified by data set and the variable q_0, which determines the size of the core tensor W used in the Tucker decomposition. We recall that a value of $q_0 = 1$ reproduces the Kronecker structured mean as considered earlier.

We know that the computational cost will scale as $O(pq_0^{p+1}) + O(pm_0q_0^2)$, so we would expect the timing data to reflect this appropriately for each data set, with $p = 2$, $p = 4$ and $p = 8$ for each data set respectively.

Method	Test RMSEs	q_0	Data set: Jutland	Data set: CCPP	Data set: Household
TVGP	Means ± stds	1	0.865 ± 0.0533	$\mathbf{0.563} \pm 0.664$	$\mathbf{1.16} \pm 2.08$
		2	0.831 ± 0.0815	0.997 ± 1.99	7.83 ± 26.3
		3	$\mathbf{0.822} \pm 0.0838$	2.45 ± 5.26	110.0 ± 773.0
	Medians ± iqrs	1	0.874 ± 0.118	0.361 ± 0.188	0.819 ± 0.402
		2	0.851 ± 0.185	$\mathbf{0.309} \pm 0.718$	$\mathbf{0.64} \pm 1.29$
		3	$\mathbf{0.839} \pm 0.189$	0.398 ± 1.18	0.83 ± 2.86
SVIGP	Means ± stds	$m = 10$	0.723 ± 0.024	0.283 ± 0.0108	$\mathbf{0.0489} \pm 0.0127$
		$m = 50$	$\mathbf{0.678} \pm 0.0555$	$0.278 \pm 8.02\text{e}-3$	0.105 ± 0.0448
		$m = 200$	$0.992 \pm 1.8\text{e}-7$	0.296 ± 0.117	0.287 ± 0.236
		$m = 500$	$0.992 \pm 1.93\text{e}-8$	$\mathbf{0.276} \pm 5.22\text{e}-3$	0.461 ± 0.251
	Medians ± iqrs	$m = 10$	0.711 ± 0.0288	$0.278 \pm 7.88\text{e}-3$	$\mathbf{0.0449} \pm 0.0195$
		$m = 50$	$\mathbf{0.658} \pm 0.0889$	$0.275 \pm 6.33\text{e}-3$	0.0954 ± 0.0727
		$m = 200$	$0.992 \pm 8.59\text{e}-9$	0.275 ± 0.0151	0.161 ± 0.349
		$m = 500$	$0.992 \pm 1.79\text{e}-9$	$\mathbf{0.274} \pm 4.6\text{e}-3$	0.57 ± 0.522

Table 2.2: The test RMSEs of the TVGP, stratified by data set and q_0, compared to the test RMSEs of the SVIGP, stratified by data set and m.

In Table 2.2, we observe a very significant difference between the means and the medians for the test RMSE of the TVGP, especially for the data sets of larger dimensionality. Increasing from $q_0 = 1$ to $q_0 = 2$ improves the results, but increasing further to $q_0 = 3$ only improves the test RMSE for the Jutland data set. It is likely that increasing the number of variational parameters can make the inference more challenging, resulting in incomplete optimisation and a high test RMSE.

Method	Times (s)	q_0	Data set: Jutland	Data set: CCPP	Data set: Household
TVGP	Means ± stds	1	**179** ± 50.7	**307** ± 114	**775** ± 327
		2	196 ± 65.4	383 ± 168	1.68e3 ± 1.30e3
		3	218 ± 77.9	486 ± 262	2.12e4 ± 2.91e4
	Medians ± iqrs	1	**163** ± 72.0	**267** ± 152	**660** ± 428
		2	173 ± 85.0	323 ± 230	1.16e3 ± 1.12e3
		3	192 ± 106	389 ± 336	9.14e3 ± 2.21e4
SVIGP	Means ± stds	$m = 10$	**187.0** ± 18.9	**195.0** ± 34.6	**188.0** ± 28.2
		$m = 50$	203.0 ± 24.7	198.0 ± 24.1	201.0 ± 24.2
		$m = 200$	779.0 ± 94.6	846.0 ± 152.0	824.0 ± 105.0
		$m = 500$	8.2e3 ± 5.37e3	6.28e3 ± 327.0	6.56e3 ± 374.0
	Medians ± iqrs	$m = 10$	**190.0** ± 21.8	**204.0** ± 46.8	**186.0** ± 54.0
		$m = 50$	202.0 ± 51.8	193.0 ± 33.9	194.0 ± 34.4
		$m = 200$	776.0 ± 207.0	843.0 ± 241.0	796.0 ± 238.0
		$m = 500$	6.7e3 ± 967.0	6.22e3 ± 576.0	6.62e3 ± 488.0

Table 2.3: The computational times of the TVGP optimisation, stratified by data set and q_0, compared to the computational times of the SVIGP, stratified by data set and m.

By contrast, in the timing results in Table 2.3, the difference between the means and medians is not so pronounced, and suggests a minor skew in the opposite direction than the test RMSE results. We observe that the computational time does increase with q_0, but not with as extreme a scaling as would be suggested by the theoretical $O(q_0^p)$ scaling. It is likely that other computations contribute rate-limiting steps to the algorithm, weakening the empirical scaling with q_0.

2.5.4.3 Dependence of Results on q_0 and m_0

Here we consider results stratified by the data set, the Tucker rank q_0, and m_0, the size of the inducing point grid in each dimension. Again we analyse the means, standard deviations, medians and interquartile ranges of the statistics of interest. The tables of the data stratified to this level are quite bulky, so are presented in Appendix A

The test RMSE results in Tables A.1 and A.2 demonstrate an interaction between q_0 and m_0: the optimisation is generally successful with a small value of m_0 and a large value of q_0, or vice versa, but the extremely large outliers are achieved when both q_0 and m_0 are large. We observe that the robust median statistic of the results also becomes very large in this regime, suggesting that this is a systemic property of many of the results, and not the effect of a few extreme outliers. This effect is most pronounced for the data with large dimensionality, which will have far higher numbers of variational parameters for larger q_0 and m_0.

We would theoretically anticipate scaling with m_0, q_0 and p of the form $O(pq_0^{p+1})+O(pm_0q_0^2)+O(pm_0^3)$. The observed timing results reflect this expected behaviour: In Tables A.3 and A.4, we see computational time increasing with m_0 and q_0 in for all of the results, with more marked

increase (especially of q_0) in the higher-dimensional data sets. This suggests that the computations scaling with m_0 are rate-limiting in lower dimensions, and the computations with q_0 are rate-limiting in higher dimensions, which is sensible given the theoretical scaling.

The difference between the median and mean values suggests a positive skew in the timing results, and the very large values of the standard deviation and interquartile ranges for the more expensive runs suggests a very broad distribution of results.

2.5.4.4 Dependence of Results on q_0 and b

It is also of interest to consider the scaling of the algorithm with the data set, the Tucker rank q_0, and the batch size b. Increasing b will have the effect of reducing the variance of the estimates of the gradient. We would expect a scaling in computational cost of the form $O(pbm_0^2)$, as the summing over the batch is a linear operation. The results tables can again be found in Appendix A.

We see from Tables A.5 and A.6 that increasing the size of the batch in general decreases the test RMSE. The highest extreme values of the mean test RMSE come with small values of b and $q_0 = 3$ in the high-dimensional Household data set. Consequently, it seems that using larger batch sizes has a positive effect when trying to optimise a heavily parametrised variational distribution.

We see from Tables A.7 and A.8 that in general computational times weakly increase with batch size, suggesting that the expected scaling is true, but not the rate-limiting step. An interesting exception is for $q_0 = 2$ and $q_0 = 3$ for the Household data set, in which the computational times hugely increase for larger b. It is surprising that the batch computation would be the rate-limiting step in this context, considering the large and separate operations performed on the variational core tensor W.

2.5.4.5 Best Performing Experimental Conditions

In Tables 2.4 we present the combinations of q_0, m_0 and b that produce the lowest values of test RMSE for each data set when averaged over the random seeds. These should give some insight into the best values of these variables to use for the Tucker Variational GP within the optimisation framework used.

Data	b	q_0	m_0	Test RMSE	Time (s)	Log Bound
Jutland	100	3	10	0.713 ± 0.0138	242 ± 0.712	$-3.25e6 \pm 4.07e5$
	10	3	10	0.726 ± 0.0261	205 ± 0.564	$-3.22e6 \pm 5.63e5$
	50	3	10	0.726 ± 0.0352	246 ± 2.88	$-3.36e6 \pm 6.81e5$
	100	2	10	0.732 ± 0.0167	219 ± 0.605	$-3.38e6 \pm 4.07e5$
	10	2	10	0.739 ± 0.0238	186 ± 1.07	$-3.6e6 \pm 1.01e6$
CCPP	10	3	5	0.25 ± 0.0834	260 ± 86.7	$-9.27e3 \pm 5.46e3$
	100	3	5	0.272 ± 0.00207	396 ± 1.27	$-2.36e3 \pm 1.27e3$
	100	3	3	0.273 ± 0.000869	305 ± 1.26	$-8.22e3 \pm 916.0$
	50	2	5	0.273 ± 0.00352	284 ± 1.23	$-2.76e3 \pm 2.12e3$
	10	2	5	0.273 ± 0.00397	258 ± 1.54	$-8.54e3 \pm 1.67e3$
Household	100	2	3	0.092 ± 0.01	1170 ± 4.57	$-2.06e6 \pm 8.65e4$
	50	2	3	0.106 ± 0.00893	848 ± 1.64	$-2.54e6 \pm 1.77e5$
	10	2	3	0.12 ± 0.02	585 ± 0.974	$-3.21e6 \pm 3.88e5$
	100	2	5	0.139 ± 0.0432	1650 ± 5.79	$-1.58e6 \pm 1.34e6$
	50	2	5	0.15 ± 0.041	1150 ± 2.36	$-1.85e6 \pm 8.02e5$

Table 2.4: The best performing experimental conditions for the TVGP, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs, times in seconds and final evaluations of the log variational bound.

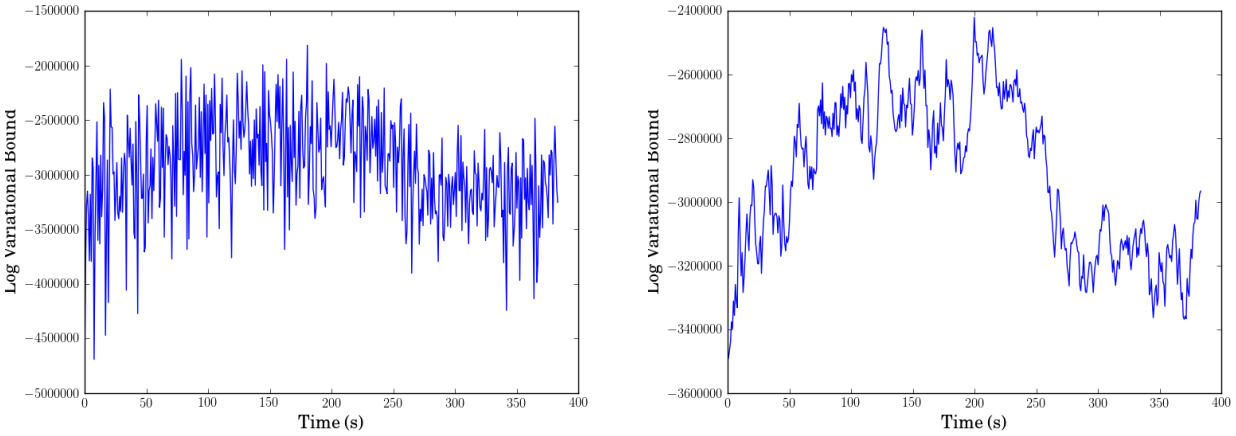
We see that for the 2 dimensional Jutland data set, the best performing runs have larger values of q_0, m_0 and b: this is sensible, considering that the smaller dimensionality will give in a smaller total number $q_0^p + pm_0q_0$ of variational parameters, resulting in a tractable inference problem. We also observe that the final evaluation of the variational bound generally increases with predictive ability.

For the 4 dimensional CCPP data set, we observe that smaller values of m_0 produce the best results, although keeping b and q_0 large still appears to be optimal. As we have increased the dimension, it seems likely that large values of m_0 result in inferentially intractable numbers of variational parameters, consequently reducing overall performance. There is no clear link between predictive ability and the final variational bound value

Finally, for the 8 dimensional Household data set, we observe that smaller values of both m_0 and q_0 are best, but maintaining a large b is still optimal. It is again likely that the larger value of p results in huge numbers of variational parameters with the biggest values of q_0 and m_0, resulting in a very challenging optimisation. We again do not observe a clear relationship between the optimised variational bound value and the test RMSE.

We observe that when averaged over random seeds, these results have quite small standard deviations for the RMSE and the times, suggesting that the best performing regimes exhibit consistent inference and the large positive skew and outliers encountered earlier are absent.

It is also of interest to consider the best performing individual runs for each data set, with the progression of the value of the variational bound every 100 iterations over the course of optimisation. These are presented in unsmoothed form, and also smoothed over a window of 10 values, in Figures 2.5, 2.6 and 2.7. The variational bound was evaluated stochastically using batches of the data, contributing some noise to the values observed.



(a) Unsmoothed (b) Smoothed

Figure 2.5: The TVGP run with lowest test RMSE on the Jutland data set, with $m_0 = 20$, $q_0 = 3$, $b = 100$ and random seed equal to 8. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.681 and took a time of 384 seconds to optimise.

We can see that the best CCPP and Household follow a trajectory starting from a poor initialisation and eventually converging, with some stochasticity, to a mode. By contrast, the best Jutland run follows a slightly less clear optimisation path, not apparently converging to the largest value explored. This is quite surprising: it is perhaps an demonstration that gradient descent algorithms are not necessarily robust enough to ensure completely robust convergence in practice, especially given the very high variance observed on the variational bound evaluations in Figure 2.5.

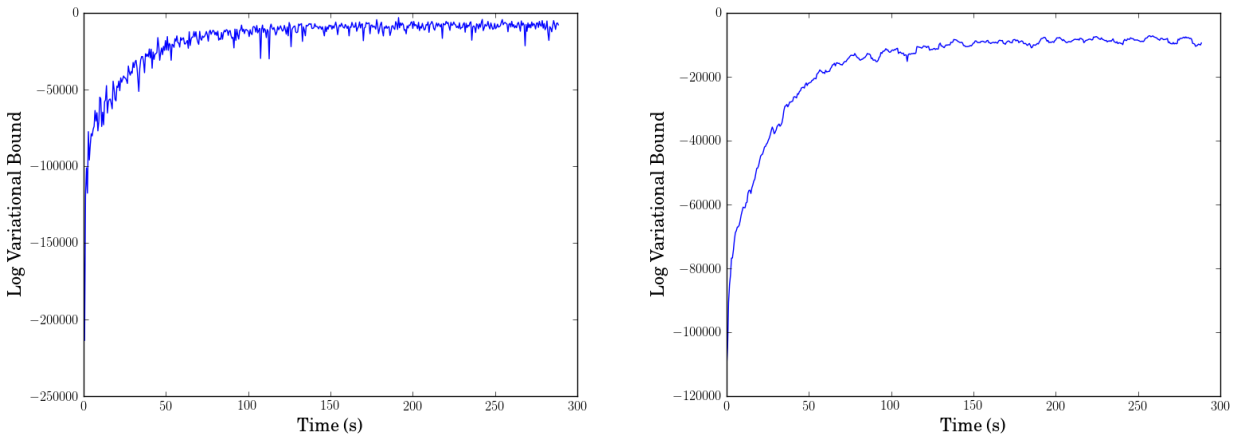
We also see that the best run for the Jutland data set has the largest possible number of variational parameters and batch size, which is likely a result of the small number of dimensions keeping the inference tractable.

By contrast, the best run for the CCPP data set has moderate values of m_0 and q_0, which should generate a manageable number of total variational parameters, given the slightly larger dimensionality. Yet further, we see that the best run for the data set Household has a value of $q_0 = 1$, reproducing the Kronecker structure discussed earlier. This has the disadvantage of being a less flexible variational representation, but the advantage of generating far fewer parameters to optimise.

2.6 Comparison with SVIGP

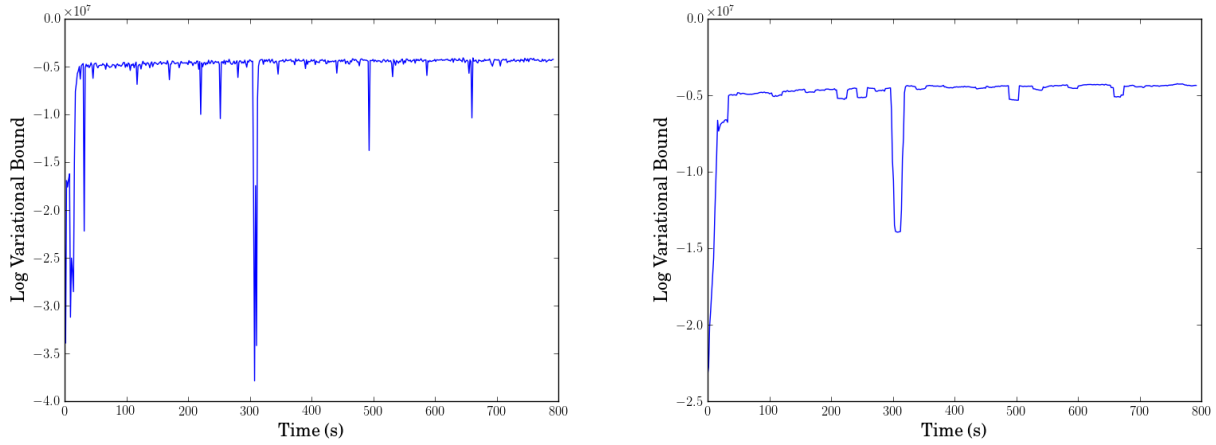
2.6.1 SVIGP Experimental Results

Here we discuss the results of the experiments using the GPy SVIGP, presented stratified across data sets, m and b. The means, medians, standard deviations and interquartile ranges across runs of the test RMSE and the computational times are presented. These statistics



(a) Unsmoothed (b) Smoothed

Figure 2.6: The TVGP run with lowest test RMSE on the CCPP data set, with $m_0 = 5$, $q_0 = 3$, $b = 10$ and random seed equal to 9. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.268 and took a time of 289 seconds to optimise.



(a) Unsmoothed (b) Smoothed

Figure 2.7: The TVGP run with lowest test RMSE on the Household power consumption data set, with $m_0 = 10$, $q_0 = 1$, $b = 50$ and random seed equal to 7. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.0508 and took a time of 793 seconds to optimise.

should demonstrate the predictive power of the model, while also showing the consistency of the inference algorithm. The tables of results can be found in Appendix A.

Observing the test RMSEs in Tables A.9 and A.10, some clear trends emerge. There is generally no large difference between the mean and median of the results. The inference appears to be much more consistent, with very small deviation between different random seeds, and no extreme very large values of the test RMSE such as were observed with the TVGP results.

Increasing the batch size has an almost uniformly positive effect on the test RMSE. In future work, it would be worth exploring larger values of b in order to see at what value such improvements might asymptote.

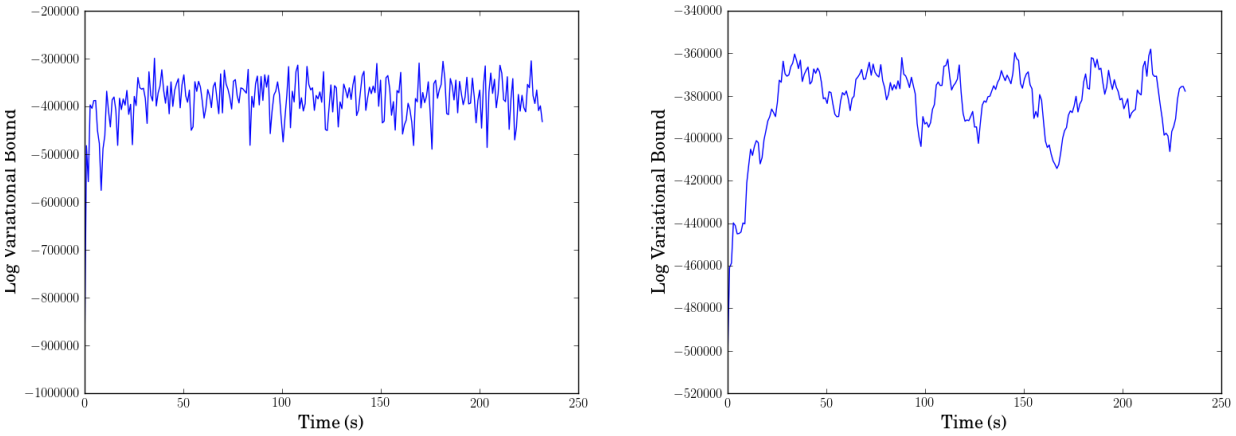
The variation of the test RMSE with m is more complex. We see that in many circumstances increasing m does not improve the performance of the model, most notably in the Jutland data set, where all predictions with $m > 50$ are uninformative. The Jutland and Household data sets reach optimal performance at $m = 50$, whereas there is little substantial improvement above $m = 10$ for the CCPP data set.

Some trends are clear in the computational times shown in Table A.11 and A.12. The means and medians are very similar, and the standard deviations and interquartile ranges are very small, suggesting a consistent inference procedure. There is one exception, when using the Jutland data set and $b = 1$ with $m = 500$, for which optimising a large variational distribution with very small batches is clearly quite challenging.

We observe an increase in computational time with m, which would sit well with our $O(m^3)$ expectations, and also an increase in time with b. We observe that the increase in computational time with b becomes less marked as m becomes large, suggesting that the $O(m^3)$ computations become the rate-limiting step in this regime.

Data	m	b	Test RMSE	Time (s)	Log Bound
Jutland	50	100	$0.624 \pm 9.12\text{e}-3$	232 ± 0.691	$-3.93\text{e}5 \pm 3.67\text{e}4$
	50	50	$0.641 \pm 9.27\text{e}-3$	203 ± 0.378	$-3.78\text{e}5 \pm 4.44\text{e}4$
	50	10	0.683 ± 0.0149	180 ± 0.722	$-5.34\text{e}5 \pm 3.02\text{e}5$
	10	100	$0.704 \pm 3.22\text{e}-3$	203 ± 16.9	$-4.35\text{e}5 \pm 4.94\text{e}4$
	10	50	$0.707 \pm 2.4\text{e}-3$	192 ± 9.61	$-4.15\text{e}5 \pm 5.88\text{e}4$
CCPP	200	100	$0.269 \pm 1.75\text{e}-3$	970 ± 40.9	$-423. \pm 1.91\text{e}3$
	200	50	$0.272 \pm 2.14\text{e}-3$	937 ± 69.2	$-54.7 \pm 1.0\text{e}3$
	50	100	$0.273 \pm 1.18\text{e}-3$	235 ± 2.26	36.5 ± 386.0
	50	50	$0.274 \pm 8.69\text{e}-4$	204 ± 0.481	103.0 ± 814.0
	10	100	$0.276 \pm 7.8\text{e}-4$	212 ± 8.02	$-672. \pm 1.27\text{e}3$
Household	10	100	$0.0396 \pm 6.24\text{e}-3$	188 ± 25.2	$2.92\text{e}6 \pm 1.91\text{e}6$
	10	50	$0.0403 \pm 4.71\text{e}-3$	207 ± 17.8	$3.72\text{e}6 \pm 2.08\text{e}5$
	10	10	0.0536 ± 0.0147	204 ± 17.8	$3.25\text{e}6 \pm 7.85\text{e}5$
	50	100	$0.0542 \pm 6.63\text{e}-3$	238 ± 0.754	$1.18\text{e}6 \pm 1.64\text{e}6$
	10	1	$0.0619 \pm 4.64\text{e}-3$	154 ± 10.6	$3.62\text{e}6 \pm 3.0\text{e}5$

Table 2.5: The best performing experimental conditions for the SVIGP, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs, times in seconds and final evaluations of the log variational bound.



(a) Unsmoothed (b) Smoothed

Figure 2.8: The SVIGP run with lowest test RMSE on the Jutland data set, with $m = 50$, $b = 100$ and random seed equal to 0. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.614 and took a time of 233 seconds to optimise.

The best performing experimental values, averaged over random seed are presented in Table 2.5. We see different conditions are preferred between different data sets. The Jutland data set is best modelled using a moderate number of inducing points and a large batch size, the CCPP data set is best modelled with a slightly larger m and large b, and the Household data set is surprisingly best modelled with a very small number of inducing points and a large batch size.

The individual best runs for each data set can be seen in Figures 2.8, 2.9 and 2.10. We see that each of the best runs is drawn from the optimally performing experimental conditions observed in Table 2.5, and all follow a fairly clear schedule of starting from a poorly initialised position and converging to a stable, stochastically evaluated, optimum.

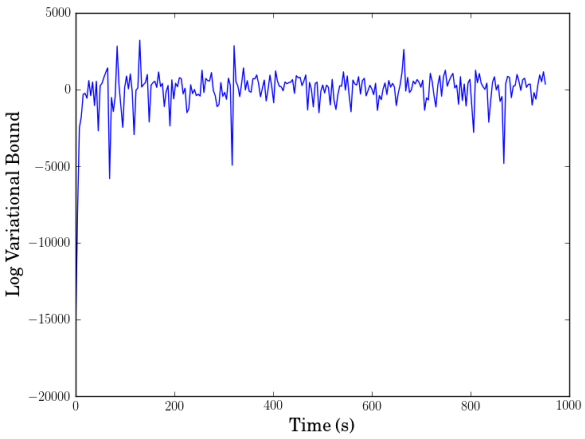
2.6.2 Comparison of Tucker Variational GP and SVIGP Experimental Results

When we compare the best performing seed-averaged experimental regimes for the TVGP and SVIGP methods, we see that the best SVIGP runs outperform the TVGP runs in both RMSE and time for the Jutland and Household data sets, while for the CCPP data set the methods achieve very similar test RMSEs but the TVGP has more preferable computational times.

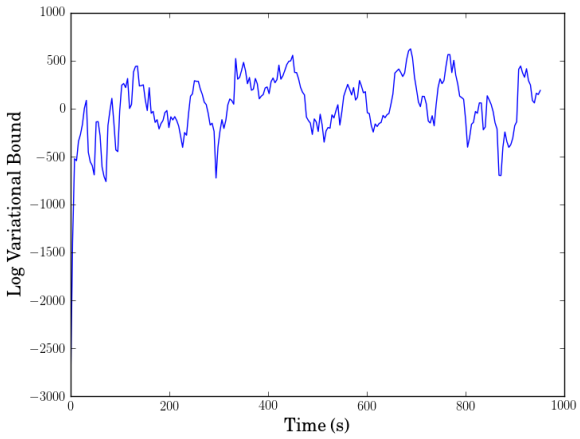
Similarly, we can see in Figure 2.11 that for the CCPP data set, the Tucker Variational GP and the SVIGP jointly occupy the leading curve describing the trade-off of performance and time, whereas, for the other two data sets, the TVGP does not clearly exhibit a regime in which it might be preferred to the SVIGP.

2.6.2.1 Hyperparameter Comparison

It is of interest to compare the optimised ARD length scales for the Tucker Variational GP and the SVIGP. In Table 2.6 we present the optimised length scales for the runs of each model with

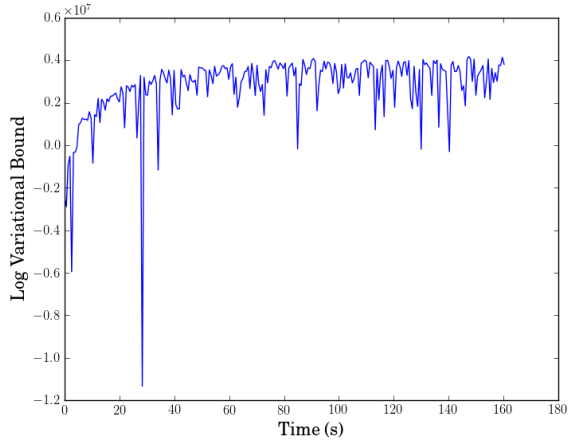


(a) Unsmoothed

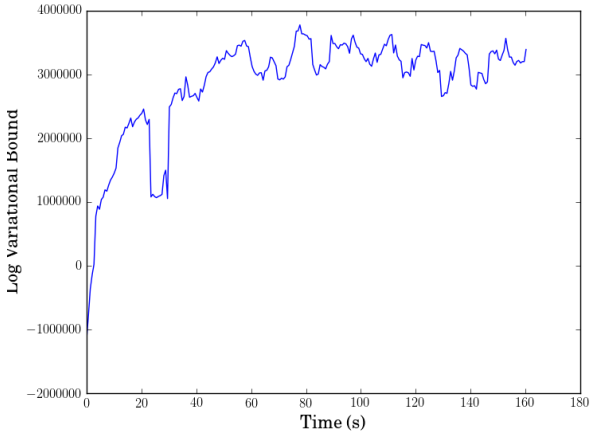


(b) Smoothed

Figure 2.9: The SVIGP run with lowest test RMSE on the CCPP data set, with $m = 200$, $b = 100$ and random seed equal to 4. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.264 and took a time of 955 seconds to optimise.

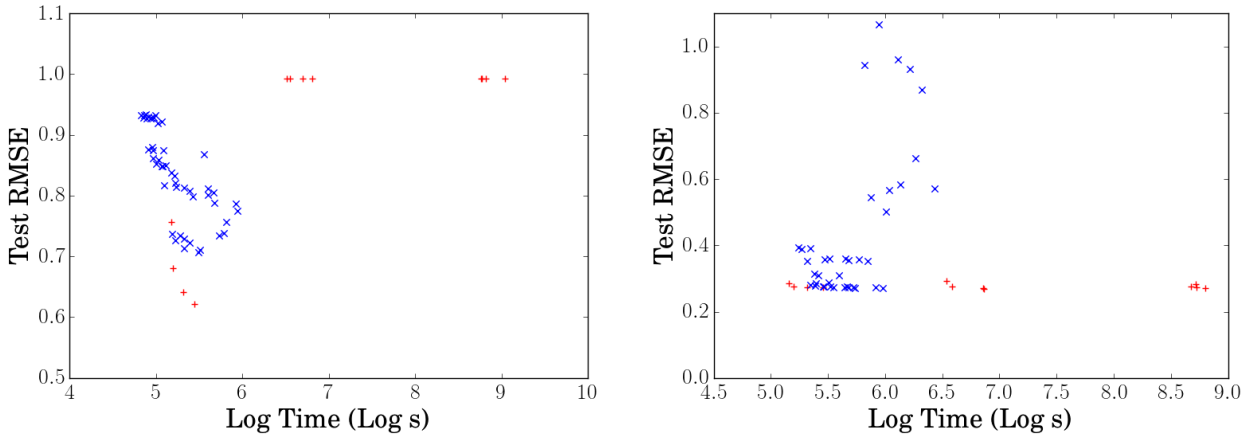


(a) Unsmoothed



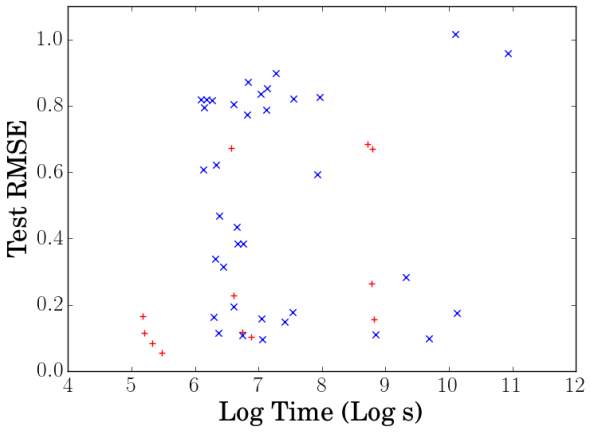
(b) Smoothed

Figure 2.10: The SVIGP run with lowest test RMSE on the Household power consumption data set, with $m = 10$, $b = 100$ and random seed equal to 7. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.0343 and took a time of 161 seconds to optimise.



(a) Jutland

(b) CCPP



(c) Household

Figure 2.11: Scatter plots of test RMSE medians and log time medians taken across random seeds. SVIGP results are presented as plus signs, and TVGP results as crosses.

the lowest test RMSE on the CCPP data set.

p	TVGP l_p	SVIGP l_p
1	0.913	1.599
2	0.873	0.580
3	1.294	5.218
4	1.119	9.147

Table 2.6: The optimised ARD length scales of each dimension for the TVGP and SVIGP runs with lowest test RMSE on the CCPP data set, i.e. the TVGP run with $m_0 = 5$, $q_0 = 3$, $b = 10$ and random seed equal to 9, and the SVIGP with $m = 200$, $b = 100$ and random seed equal to 4.

We see in Table 2.6 that both methods pick out the first two dimensions as the most powerful kernel components, assigning them the shortest length scales. The length scales for the other two dimensions found by the SVIGP are much larger than those for the TVGP, suggesting that the SVIGP finds these much less useful as predictors. It is possible that the different variational parametrisation of the TVGP means that the optimal hyperparameter values are not the same, but it is also true that the inference is much more challenging.

2.6.2.2 Scalability with p

The success of the Tucker Variational GP on the 4-dimensional CCPP data set suggests that there exists a moderate value of p for which the TVGP is best suited. For large p, the exponential scaling in q_0^p renders the computational time unfavourable, and for very small p, the commitment to a Tucker decomposition is a parametric overcommitment.

More formally, we can consider the total number of variational parameters that we can successfully optimise given the data, with $q_{\max} = q_0^p$ being the maximum tractable number of elements of the core tensor W, and $U_{\max} = pm_0q_0$ the maximum tractable number of elements of the factor matrices U. If we consider the computational complexity of the TVGP, treating the total number of mean variational parameters as a constant, we derive the following computational complexity:

$$O(pq_0^{p+1} + pm_0q_0^2 + pm_0^3) = O(pq_{\max}^{\frac{p+1}{p}} + U_{\max}q_{\max}^{\frac{1}{p}} + U_{\max}^3 p^{-2} q_{\max}^{-\frac{3}{p}}). \tag{2.22}$$

Examining the limits of the above term, we see that as $p \to 0$, the term $U_{\max}q_{\max}^{\frac{1}{p}}$ dominates, taking the whole expression to infinity, and as $p \to \infty$, the term $pq_{\max}^{\frac{p+1}{p}} \to pq_{\max}$ dominates, resulting in positive linear scaling for the computation. Between these two limits, we observe that there is a dimension with minimum computational cost for a constant number of tractable variational parameters. This minimum suggests that there exists a value of p for which the TVGP is operating with maximal computation efficiency for the size of variational distribution used.

When directly comparing the results from the TVGP and SVIGP methods, it is clear that the inference is much less consistent for the structured variational method: this could be an inherent property of the model, or it could be a result of an undertuned optimisation scheme.

Specifically, systematic use of a different SGD algorithm such as Adam (Kingma and Ba, 2014) for the Tucker Variational GP would be an instructive exercise (Ruder, 2016), as would further exploration of iteration schedules: we may be under-optimising the bound, and hence misrepresenting the trade-off between performance and time.

The GPy method's ability to optimise the inducing points represents a different solution to the problem approached by the gridded inducing points. Instead of trying to systematically cover all of covariate space using a grid, the SVIGP instead attempts to optimise its independent inducing points towards the salient regions. This is much more tractable in lower dimensions, possibly explaining its superior performance for small m in the $p = 2$ data set.

2.7 Conclusions and Future Work

In principle, it is a worthwhile ambition to attempt to maximise the number of inducing points used by a variational Gaussian Process. Further, the Tucker decomposition is an effective method of imposing structure on a tensor in order to reduce computational costs.

However, the associated inference with such model features becomes very challenging: large variational models are difficult to optimise well, and the product structure of Tucker decomposition introduces instabilities that can lead to the very extreme inaccuracies found in some of the TVGP results.

It is interesting that the TVGP competes with standard SVIGP methods on the $p = 4$ method. As discussed in Section 2.6.2.2, we might expect the gridded methods to exhibit inefficiencies for larger or very small dimensionality, so it is reasonable to see that exhibited empirically.

As discussed earlier, further work is likely necessary for tuning the optimisation of the Tucker variational bound: Exploring different update schedules and more efficient implementation would be likely to have a significant effect on the performance of the model. The inability of the TVGP to optimise the kernel and noise variance without instability also represents an optimisation problem that needs to be addressed. From the observed results, it would also appear that exploring larger values of the batch size b would be beneficial.

There is possible further theoretical work to be done as well. While the model does not scale with the number of inducing points, there is still exponential scaling in the elements of W of the form $O(q_0^p)$.

There are multiple ways to address this: the core tensor W does not need to be hypercubic, i.e. each dimension of W could have a size q_p associated with it, such that the total scaling is $O(\prod_{l=1}^{p} q_l)$. If many of the q_l are equal to 1, then the exponential scaling with p is overcome. It would make sense to assign $q_l = 1$ for dimensions that we would *a priori* expect to have less complex structure, although it is not clear what criteria would be used to make such a judgement.

Similarly, it would be possible to impose a blocky structure on W: given that the TVGP appears to work best for moderate values of $p \sim 4$, the core tensor could be blocked into segments of approximately this size. Consequently, the inference would be operating optimally within each block, and all of the dimensions would be covered. This would be at the expense of communication between blocks, which would only be able to interact through the response variable residuals.

Such an approach would be similar to running an additive model, in which individual models are run on each block and combined at the point of prediction (Hastie and Tibshirani, 1990). Previous work from this field would suggest that a blocky W approach might struggle with problems of multicollinearity and overfitting.

Alternatively, it would be possible to impose sparsity on the core tensor, such that only a constant number of elements Q are allowed to be non-zero. This would impose a strong bound on the computational complexity of $O(Q)$: it is not clear how to choose the sparsity allocations, or how this will effect the expressiveness of the variational bound.

It is conceivable that the entire project of attempting to use gridded inducing points is overambitious. The benefit of systematically covering all of covariate space with inducing points is only necessary if the data follow a space-filling process that systematically covers all of covariate space, as in the gridded data points of voxel data or similar. In these situations, it might be most efficient to develop methods that exploit the specific structure present in the data for maximum algorithmic efficiency. Alternatively, if the data do not systematically cover covariate space, then standard inducing point methods may be sufficient to target the sparsity that exists in the data.

Chapter 3

Randomised Numerics Variational Gaussian Processes

The use of randomised numerical algorithms for $O(m^2)$ evaluations of the gradients of the log variational bound of a Gaussian Process with m inducing points is pursued. A new representation of the variational distribution is introduced and a corresponding lower bound is derived. Systematic computational experiments are performed and compared with a standard variational Gaussian Process. The results are discussed, and further work is suggested. This work was a collaboration between Chris Holmes, Michalis Titsias and myself. All of the python code and all of this chapter were written by Owen Thomas.

It is known that the value of the variational bound to a Gaussian Process monotonically increases when more inducing points are added (Titsias, 2009). Consequently, it is of interest to find computationally efficient methods to increase the number of inducing points. In the spirit of previous work, here we generate unbiased approximations to the gradients of the variational bound with $O(m^2)$ computational cost, in contrast to the $O(m^3)$ necessary for the exact variational bound.

This work builds on the work of Filippone et al (Filippone and Engler, 2015), who use randomised numerical algorithms to generate unbiased estimates to the gradients of the marginal likelihood of an exact GP with complexity $O(n^2)$. Filippone et al then use these estimates to perform Stochastic Gradient Langevin Dynamics sampling over the hyperparameters (Welling and Teh, 2011). This permits computationally attractive Bayesian inference for an exact GP, without the localised biases introduced by variational methods, or resorting to the parametric representation of finite basis approximations.

Here we use similar computational approximations to evaluate the gradients of the variational bound of an inducing point framework, using a parametrisation of the variational distribution that permits $O(m^2)$ evaluations. In doing so, we can generate unbiased approximations of the variational bound, and use stochastic gradient methods to jointly optimise the variational parameters and hyperparameters.

In Section 3.1, we introduce the stochastic numerical algorithms foundational to the work. In Section 3.2, we introduce a variational distribution with $O(m)$ parameters and corresponding variational bound necessary to pursue the work. In Section 3.3 we describe the experiments

performed and present the results. Section 3.4 compares the RNVGP results with a standard variational Gaussian Process. Section 3.5 concludes and discusses potential future work.

3.1 The Randomised Algorithms

There are two randomised algorithms we will use to cheaply evaluate the relevant derivatives of the log variational bound, developed in (Filippone and Engler, 2015). The first is a method for terminating a conjugate gradient method for linear systems, such that it returns unbiased results within a few iterations. The second is a method for evaluating traces of products of matrices in $O(m^2)$ time, while introducing some unbiased noise to the value of the result.

3.1.1 Unbiased Conjugate gradients

Statisticians are often concerned with guaranteeing the unbiasedness of estimates of the quantities used in their inference procedures: many stochastic gradient descent methods assume the unbiasedness of the noisy gradients. One contribution from (Filippone and Engler, 2015) is the description of an early stopping criterion for conjugate gradient solvers, such that the result constitutes an unbiased estimate of the solution. We reproduce the algorithm here.

Standard conjugate gradient methods are used for solving the m-dimensional linear system $\mathbf{K}s = b$, by considering the answer as the minimizer of $L(s)$, defined as:

$$L(s) = \frac{1}{2}s^T\mathbf{K}s - s^T b. \tag{3.1}$$

We observe that evaluations of $L(s)$ and gradients of $L(s)$ with respect to s demand $O(m^2)$ computational cost. The optimisation problem can be approached using standard conjugate gradient optimisation methods, which will converge within m iterations, resulting in total complexity of $O(m^3)$. However, in practice, the algorithm will converge to an accurate solution within only a few iterations, if the problem is numerically well-conditioned. Consequently, it would be useful to develop an early-termination criterion for the optimisation schedule.

We can consider the solution $\mathbf{s}$ of the conjugate gradient solver up to a convergence criterion ϵ as a sum of an initial value $\mathbf{s}_0$ and T incremental terms δ_i generated by consecutive steps of a conjugate gradient optimisation:

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{T} \delta_i. \tag{3.2}$$

For computational reasons, we would like to partially evaluate this sum, with a guarantee that we will evaluate the first L terms as defined by an early stopping criterion $\epsilon_L > \epsilon$. We can rewrite the solution as a sum over the first L terms, plus a series of terms which we would like to evaluate with some probability. We introduce a set of coefficients w_r:

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{L-1} \delta_i + \frac{1}{w_0}\Big(w_0\delta_{L+0} + \frac{1}{w_1}\Big(w_0 w_1 \delta_{L+1} + \frac{1}{w_2}\Big(w_0 w_1 w_2 \delta_{L+2} + ...\Big)\Big)\Big), \tag{3.3}$$

where set of coefficients w_r can be defined as $w_r = \exp(\beta r)$, for positive β.

The algorithm then proceeds as follows: Set the initial values of $\tilde{\mathbf{s}}$ to $\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{L-1} \delta_i$. Iterating over incremental values of j, sample a uniform random variable $u_j \sim \mathcal{U}[0, 1]$. If $u_j < \frac{1}{w_j}$, then compute δ_{L+j}, set $\tilde{\mathbf{s}} = \tilde{\mathbf{s}} + \prod_{r=0}^{j} w_r \delta_{L+j}$ and increment the value of j: else return $\tilde{\mathbf{s}}$ and terminate the algorithm.

The algorithm will generate an unbiased approximation to the solution $\mathbf{K}^{-1}b$, with a guaranteed accuracy of ϵ_L, and a further accuracy up to ϵ from extra terms defined by the decay rate β. Calculating each iteration demands $O(m^2)$ computation, so the overall complexity of the algorithm will be quadratic in m, and linear in the number of terms evaluated.

A linear system is considered poorly conditioned if there is a large "condition number": the ratio of the largest eigenvalue of $\mathbf{K}$ to the smallest. It is possible in poorly conditioned systems for the number of iterations necessary for convergence to approach the size of the system m. It may also happen that a large number of iterations will introduce cumulative numerical instability that delays or prevents convergence. It is consequently necessary to pick ϵ_L and ϵ appropriately: a trade off between speed and accuracy is clear, but numerical errors will mean that arbitrarily small errors cannot be achieved for ill-conditioned systems.

3.1.2 Randomised Trace of Products of Matrices

When evaluating the derivatives of the marginal likelihoods of Gaussian Process models, it is common to encounter terms of the following form:

$$\mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\Big). \tag{3.4}$$

We observe that to evaluate the above exactly, it is necessary to perform a computation of $O(m^3)$. (Filippone and Engler, 2015) also used a method that can compute an unbiased estimate of this expression in $O(m^2)$. We introduce a matrix $\mathbf{r}$ of size $(m \times d)$, the entries of which are sampled from independent distributions with mean zero and variance 1, e.g. standard normal distributions, or Bernoulli distributions. The expectation of the matrix $\frac{\mathbf{r}\mathbf{r}^T}{d}$ is an identity matrix of size $(m \times m)$, i.e. $\mathbb{E}[\frac{\mathbf{r}\mathbf{r}^T}{d}] = \mathbb{I}$.

It is possible to use the matrix $\mathbf{r}$ to generate unbiased estimates of traces of products of matrices. Consider the expectation of the trace with the inclusion of the term $\frac{\mathbf{r}\mathbf{r}^T}{d}$ into the product inside the trace:

$$\begin{aligned}\mathbb{E}\Big[\mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\frac{\mathbf{r}\mathbf{r}^T}{d}\Big)\Big] =& \mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\mathbb{E}\Big[\frac{\mathbf{r}\mathbf{r}^T}{d}\Big]\Big) \\ =& \mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\mathbb{I}\Big) \\ =& \mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\Big).\end{aligned} \tag{3.5}$$

We see that the expectation of the entire trace remains unaffected. However, it is further possible to cyclically permute the terms in the unbiased approximation to reduce computational costs:

$$\mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\ \frac{\mathbf{r}\mathbf{r}^T}{d}\Big) = \frac{1}{d}\mathrm{tr}\Big(\mathbf{r}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\ \mathbf{r}\Big). \tag{3.6}$$

We now notice that if we use the above described $O(m^2)$ conjugate gradient solver, and evaluate the above expression from the outsides in, we will only encounter computations of $O(m^2)$. The method will also scale with $O(d)$, as we are effectively taking the mean of d independent approximations of the trace.

We also observe that by writing a component of the bound derivative in terms of $\frac{\partial \mathbf{K}}{\partial \theta}$, then we are implicitly using forward mode differentiation. This is a mode of differentiation in which the derivative of the full bound F with respect to the parameter θ is written as a product using the chain rule: the product is then evaluated starting from the final derivative with respect to θ, evaluated until the full derivative of the bound is achieved.

For example, if the bound $F(g(\theta))$ can be written in terms of an interior function $g(\theta)$, then we can write down the full derivative $\frac{\partial F}{\partial \theta}$ using the chain rule:

$$\frac{\partial F}{\partial \theta} = \frac{\partial F}{\partial g}\frac{\partial g}{\partial \theta}. \tag{3.7}$$

In forward mode differentiation, the derivative $\frac{\partial g}{\partial \theta}$ is first evaluated, then multiplied with $\frac{\partial F}{\partial g}$ to calculate the full expression.

By contrast, it is also possible to pursue reverse-mode differentiation, in which the product of derivatives produced by the chain rule is evaluated from the first derivative of the bound F until the derivative with respect to the parameter θ is encountered. In this situation, the derivative $\frac{\partial F}{\partial g}$ would first be evaluated, then multiplied with the term $\frac{\partial g}{\partial \theta}$.

The distinction between forward and reverse mode differentiation is most acutely observed when performing automatic differentiation: certain computational inference packages are able to symbolically differentiate model likelihoods or losses, and it is in the interests of those designing these programs to ensure that this operation is done as efficiently as possible. For this project, the derivatives were encoded by hand, such that the gradient evaluations were hand-tuned to be efficient for this problem, at the cost of greater implementation time.

3.2 An $O(m)$ Variational Parametrisation

It is not enough to introduce the randomised numerics to the general stochastic variational bound to achieve $O(m^2)$ computation: evaluations of the gradient with respect to the standard variational covariance will be $O(m^3)$, in a way that cannot be reduced by the methods described above. Specifically, the term $-\frac{1}{2}\log|\mathbf{S}|$ in Equation 1.35 has a derivative with respect to $\mathbf{S}$ of $-\frac{1}{2}\mathbf{S}^{-1}$: none of the randomised numerical methods described above are capable of reducing the computational cost of this gradient term to $O(m^2)$.

Some reparametrisation is necessary: specifically, if we can introduce an $O(m)$ parametrisation of the covariance, then gradient evaluations that are $O(m^2)$ in computation should become viable. Having experimented with some rank-reduced representations of the covariance, we derived a variational distribution with $2m$ parameters for the latent distributions evaluated at the inducing points of the form:

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{K}_{\mathbf{uu}}\boldsymbol{\mu}, \mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(\mathbf{K}_{\mathbf{uu}} + \Sigma)^{-1}\mathbf{K}_{\mathbf{uu}}), \tag{3.8}$$

or equivalently:

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{K}_{\mathbf{uu}}\boldsymbol{\mu}, (\mathbf{K}_{\mathbf{uu}}^{-1} + \Sigma^{-1})^{-1}), \tag{3.9}$$

where $\boldsymbol{\mu}$ is a $m \times 1$ vector representing the preconditioned variational mean. The covariance representation is motivated by (Opper and Archambeau, 2009): the $m \times m$ matrix Σ is diagonal, and all of its m diagonal elements are positive.

3.2.1 Variational Bound

We recall the expression for the log variational bound for the stochastic variational Gaussian Process introduced in Equation 1.35:

$$\begin{aligned} F =& \sum_{i=1}^{n}\left\{\log\mathcal{N}(y_i|\mathbf{k}_{\mathbf{f}_i\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m}, \sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}(\mathbf{S}\Lambda_i)\right\} \\ &+ \frac{1}{2}\log|\mathbf{S}| - \frac{1}{2}\log|\mathbf{K}_{\mathbf{uu}}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}\right) - \frac{1}{2}\mathbf{m}^T\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m} + \frac{m}{2}, \qquad (3.10) \\ \Lambda_i =& \sigma^{-2}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}i}\mathbf{k}_{i\mathbf{u}}^T\mathbf{K}_{\mathbf{uu}}^{-1}. \qquad (3.11) \end{aligned}$$

We now introduce the new parametrisation of the variational parameters $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}$ and $\mathbf{S} = \mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(\mathbf{K}_{\mathbf{uu}} + \Sigma)^{-1}\mathbf{K}_{\mathbf{uu}}$, i.e.:

$$\begin{aligned} F =& \sum_{i=1}^{n}\left\{\log\mathcal{N}(y_i|\mathbf{k}_{\mathbf{f}_i\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha}, \sigma^2) - \frac{1}{2}\beta\tilde{\mathbf{k}}_{ii} - \frac{1}{2}\mathrm{tr}\left((\mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(\mathbf{K}_{\mathbf{uu}} + \Sigma)^{-1}\mathbf{K}_{\mathbf{uu}})\Lambda_i\right)\right\} \\ &+ \frac{1}{2}\log|\mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(\mathbf{K}_{\mathbf{uu}} + \Sigma)^{-1}\mathbf{K}_{\mathbf{uu}}| - \frac{1}{2}\log|\mathbf{K}_{\mathbf{uu}}| \\ &- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{uu}}^{-1}(\mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(\mathbf{K}_{\mathbf{uu}} + \Sigma)^{-1}\mathbf{K}_{\mathbf{uu}})\right) - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}_{\mathbf{uu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uu}}\boldsymbol{\alpha} + \frac{m}{2}, \qquad (3.12) \\ \Lambda_i =& \sigma^{-2}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}i}\mathbf{k}_{i\mathbf{u}}^T\mathbf{K}_{\mathbf{uu}}^{-1}. \qquad (3.13) \end{aligned}$$

After some algebra, the following simplified log variational bound emerges:

$$
\begin{aligned}
F =& \log \mathcal{N}(\mathbf{y}|\mathbf{K_{fu}}\boldsymbol{\mu}, \sigma^2 I) - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K_{ff}}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K_{fu}}(\mathbf{K_{uu}}+\Sigma)^{-1}\mathbf{K_{uf}}\right) \\
& - \frac{1}{2}\boldsymbol{\mu}^T\mathbf{K_{uu}}\boldsymbol{\mu} + \frac{1}{2}\mathrm{tr}\left((\Sigma+\mathbf{K_{uu}})^{-1}\mathbf{K_{uu}}\right) - \frac{1}{2}\log|\Sigma+\mathbf{K_{uu}}| + \frac{1}{2}\log|\Sigma| \qquad (3.14) \\
=& \sum_{i=1}^{n}\left(\log \mathcal{N}(y_i|\mathbf{k_{f_iu}}\boldsymbol{\mu}, \sigma^2) - \frac{1}{2\sigma^2}\mathbf{k_{f_if_i}} + \frac{1}{2\sigma^2}\mathbf{k_{f_iu}}(\mathbf{K_{uu}}+\Sigma)^{-1}\mathbf{k_{uf_i}}\right) \\
& - \frac{1}{2}\boldsymbol{\mu}^T\mathbf{K_{uu}}\boldsymbol{\mu} + \frac{1}{2}\mathrm{tr}\left((\Sigma+\mathbf{K_{uu}})^{-1}\mathbf{K_{uu}}\right) - \frac{1}{2}\log|\Sigma+\mathbf{K_{uu}}| + \frac{1}{2}\log|\Sigma|. \qquad (3.15)
\end{aligned}
$$

We observe that the bound can be expressed as a sum over the data points, and is hence suitable for stochastic subsampling of the data for inference, in the style of (Hensman et al., 2013). At this stage, the bound is $O(m^3)$ in computation, and contains $2m$ variational parameters.

3.2.2 Derivatives of the Variational Bound

Here we present the derivatives with respect to a kernel hyperparameter θ:

$$
\begin{aligned}
\frac{\partial F}{\partial \theta} =& \frac{1}{\sigma^2}\mathrm{tr}\left(\frac{\partial \mathbf{K_{fu}}}{\partial \theta}\left(\boldsymbol{\mu}(\mathbf{y}-\mathbf{K_{fu}}\boldsymbol{\mu})^T + (\mathbf{K_{uu}}+\Sigma)^{-1}\mathbf{K_{uf}}\right)\right) - \frac{1}{2\sigma^2}\mathrm{tr}\left(\frac{\partial \mathbf{K_{ff}}}{\partial \theta}\right) \qquad (3.16) \\
& - \frac{1}{2}\mathrm{tr}\left(\frac{\partial \mathbf{K_{uu}}}{\partial \theta}\left((\mathbf{K_{uu}}+\Sigma)^{-1}(\mathbf{K_{uf}}\mathbf{K_{fu}}\sigma^{-2}+\mathbf{K_{uu}})(\mathbf{K_{uu}}+\Sigma)^{-1} + \boldsymbol{\mu}\boldsymbol{\mu}^T\right)\right)
\end{aligned}
$$

We notice that each of the $O(m^3)$ terms in the derivatives are either products of matrices within traces or solutions of linear systems. As such, they are amenable to the randomised numerics described earlier.

Consequently, we are able to perform inference for the model using Stochastic Gradient Descent for optimisation, or Stochastic Gradient Langevin Dynamics for sampling. Given that we are performing joint inference over hyperparameters and variational parameters, it is easiest to use Stochastic Gradient Descent for all variables and find MAP estimates for hyperparameters. We call this the Randomised Numerics Variational Gaussian Process (RNVGP).

3.3 Experimental Results

3.3.1 Experimental Design

3.3.1.1 RNVGP Experimental Design

There are multiple properties of the approximations and inference that can be varied in simulations. Here are presented the different values assigned during the runs of the algorithm: experiments were performed systematically over every combination of these values.

Firstly, the data set was varied. The Boston housing data set was used, with $n = 506$ data points and $p = 13$ covariates. This was a good example of a small n data set to test the properties of the algorithm without encountering the problems associated with very large

numbers of observations. The data set SARCOS was also used, with $n = 44,484$ training examples, and $p = 21$ input covariates. This data set represented a more challenging inference problem, with far more data points and more covariates to include in the model. The data sets were partitioned such that 90% of the data were used for training and 10% for testing predictions, and normalised such that the covariates and response variable each have zero mean and unit variance. Consequently, test RMSEs will be reported as a consistent measure of explained variance.

Secondly, m, the number of inducing points, was varied, with values of 50, 200 and 500 used for the SARCOS data, and 50, 200 and 455 for the Boston data. We would expect a model with more inducing points to be more expressive, but contain more variational parameters and hence pose a more challenging inference problem. The difference in the largest value was because the Boston data set had 455 total training data points, so having more than this would be redundant.

Thirdly, b, the batch size of the stochastic inference algorithm, took values of 1, 10, 50 and 100. A smaller batch size will correspond to a larger variance for the estimates of the gradients.

Fourthly, d, the dimension of the matrix used to evaluate the randomised trace of the product of matrices, took values of 1, 5 and 10. We would expect a smaller value of d to produce a larger variance for the evaluations of the relevant terms.

Fifthly, the seed of the random number generator of the algorithm was given the integer values from 0 to 9. This ensured enough runs of the algorithm to enable bulk comparison of results between different experimental regimes.

Finally, the tolerances of the unbiased conjugate gradients algorithm were varied: for half of the experiments they took the values $\epsilon_L = 1\text{e}-1$ and $\epsilon = 1\text{e}-3$ and for the other half they were set to $\epsilon_L = 1\text{e}-3$ and $\epsilon = 1\text{e}-6$.

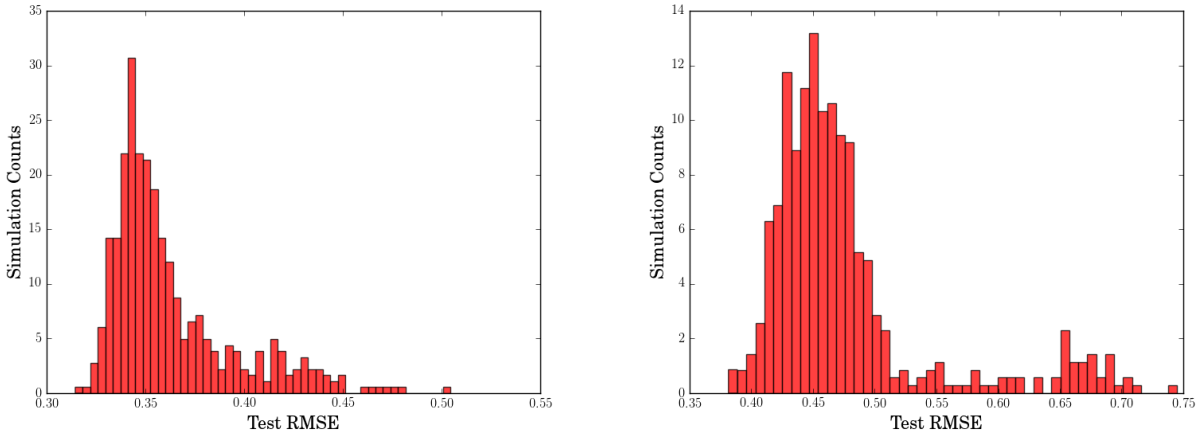
Some properties of the inference were held constant. Adagrad was used as the stochastic optimisation algorithm, and was run for 500 iterations keeping the kernel variance constant and then for $25,000$ iterations allowing all of the variables to vary freely. The GP kernel uses an ARD exponentiated quadratic covariance function.

The simulations were run on a desktop computer, with 8 cores and 7.69 GB of memory. The simulations were coded in Python, using the standard NumPy libraries for the numerics.

3.3.1.2 SVIGP Experimental Design

For comparison, we also present results run by a standard Stochastic Variational Inference Gaussian Process (SVIGP) from GPy, following the work in (Hensman et al., 2013), run on the Boston and SARCOS data sets. It should exhibit $O(m^3)$ computational complexity, with the only stochasticity in the inference being provided by the use of batches of data for inference.

The optimisation was performed using GPy's default settings of $25,000$ iterations of the AdaDelta algorithm. The GP kernel uses an ARD exponentiated quadratic covariance function. The computations were performed on the same machine as the randomised numerics experiments, allowing for comparison of computational times.



(a) Boston (b) SARCOS

Figure 3.1: Histograms of test RMSEs from the RNVGP run on the Boston Housing and SARCOS data sets.

The SVIGP experiments varied four experimental features:

Firstly, the data set was changed, using the same Boston and SARCOS data sets used for the randomised numerics variational bound.

Secondly, the number of inducing points m took values of 10, 50, 200 and 455 for the Boston data set, and 10, 50, 200 and 500 for the SARCOS data set.

Thirdly, different values of the batch size b was explored, with values of 1, 10, 50 and 100. This should demonstrate the $O(b)$ scaling and demonstrate the robustness of the algorithm to batch sizes.

Finally, the random seed was varied between the integer values from 0 to 9, ensuring there are independent results for bulk analysis.

The bulk properties of the SVIGP results will be presented alongside the RNVGP results. A more thorough presentation of the SVIGP performance will follow, then a discussion comparing the two.

3.3.2 RNVGP Bound Results

3.3.2.1 Histogram of Test RMSEs

The inference for the RNVGP was quite challenging, with the optimisation exhibiting large dispersion. In Figure 3.1, we present histograms of all of the test RMSEs of the runs performed on data sets Boston and SARCOS, respectively. As can be seen in Figure 3.1, the algorithm frequently optimised successfully, with a large cluster of test RMSE close to 0.35, but sometimes fails to optimise properly, producing other RMSEs closer to 0.5. For this reason, the distribution is quite skewed, making the bulk properties difficult to communicate.

Similarly, we can see in Figure 3.1 that the distribution over the test RMSEs of all of the randomised numerics runs on the SARCOS data set is quite skewed. We again observe inconsistent optimisation, with a cluster at 0.65, but a large cluster of more successful results at approximately 0.45. There is no long tail of results to very large values of the test RMSE.

As a result, the mean test RMSEs are presented here with standard deviations, but the median test RMSEs are also reported with the interquartile ranges. The latter statistics should better represent where the bulk of the results are located, and indicate any skew present in the results.

3.3.2.2 RNVGP Results Stratified by m

Method	Test RMSE	m	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	50	0.37 ± 0.0343	0.501 ± 0.0833
		200	0.356 ± 0.027	$\mathbf{0.450} \pm 0.0327$
		455/500	$\mathbf{0.351} \pm 0.0277$	0.467 ± 0.0481
	Medians ± iqrs	50	0.357 ± 0.0441	0.465 ± 0.0594
		200	0.348 ± 0.0228	$\mathbf{0.449} \pm 0.0458$
		455/500	$\mathbf{0.342} \pm 0.0253$	0.463 ± 0.0622
SVIGP	Means ± stds	10	0.337 ± 0.0284	0.26 ± 0.0272
		50	0.334 ± 0.0162	$\mathbf{0.234} \pm 0.022$
		200	0.324 ± 0.02	0.237 ± 0.0354
		455	$\mathbf{0.324} \pm 0.021$	0.242 ± 0.0402
	Medians ± iqrs	10	0.331 ± 0.0175	0.256 ± 0.0479
		50	0.332 ± 0.0125	$\mathbf{0.229} \pm 0.0389$
		200	$\mathbf{0.322} \pm 0.0272$	0.230 ± 0.0584
		455	0.326 ± 0.0286	0.235 ± 0.0713

Table 3.1: The test RMSEs of the RNVGP run on the Boston and SARCOS data sets, stratified by m. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison. The maximum value of m attainable for the Boston data set was 455: the highest value used for the SARCOS data set was 500.

In Table 3.1 we see the test RMSE of the RNVGP results separated by the number of inducing points m. We observe fairly marginal but consistent improvement of test RMSE for the Boston data set with increasing m. However, the SARCOS results perform optimally for $m = 200$, with the $m = 500$ variational distribution being perhaps too large to optimise successfully.

We observe that the median test RMSEs are consistently smaller than the mean test RMSEs, suggesting a positive skew in the results. The dispersion within each cell is relatively small for the Boston data set, suggesting a fairly consistent inference, although it is considerably larger for the SARCOS data set. Comparing with the average performance of the SVIGP, we see that the mean SVIGP is slightly more successful than the best Boston RNVGP results, and much more successful than any of the RNVGP SARCOS results. It is possible that the larger dispersion in the SARCOS RNVGP test RMSEs suggests that the inference is less successful for the data set with larger n and p, which would also explain the large difference in performance with the SVIGP SARCOS experiments.

Method	Time (s)	m	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	50	**135** ± 65.7	**163** ± 73.2
		200	467 ± 210	585 ± 255
		455/500	1.92e3 ± 776	2.76e3 ± 1.12e3
	Medians ± iqrs	50	**131** ± 98.4	**153** ± 111
		200	460 ± 306	560 ± 403
		455/500	1.88e3 ± 1.24e3	2.8e3 ± 1.82e3
SVIGP	Means ± stds	10	**302** ± 7.86	**296** ± 11.8
		50	331 ± 18.4	349 ± 20.4
		200	995 ± 101.0	1.03e3 ± 120
		455	4.72e3 ± 237	6.39e3 ± 396
	Medians ± iqrs	10	**302** ± 14.0	**295** ± 23.5
		50	327 ± 27.6	345 ± 30.9
		200	969 ± 142	1.03e3 ± 161
		455	4.72e3 ± 377	6.37e3 ± 881

Table 3.2: The computational times in seconds of the RNVGP run on the Boston and SARCOS data sets, stratified by m. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison. The maximum value of m attainable for the Boston data set was 455: the highest value used for the SARCOS data set was 500.

Considering the timing results for the RNVGP stratified by m in Table 3.2, we observe that the computational time increases with m, although somewhat less strongly than the anticipated $O(m^2)$ scaling. This perhaps suggests that this is not generally the rate-limiting step in the computation.

There are relatively large dispersions in observed computation time, growing in proportion to the total computation time. We see that the $m = 50$ and $m = 200$ computational times are much faster than the average SVIGP times, whereas the RNVGP runs with larger values of m match the mean values of the SVIGP computational time.

3.3.2.3 RNVGP Results Stratified by d

Method	Test RMSE	d	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	1	0.368 ± 0.0338	0.494 ± 0.0705
		5	0.354 ± 0.0299	0.466 ± 0.0538
		10	$\mathbf{0.354} \pm 0.0263$	$\mathbf{0.459} \pm 0.0557$
	Medians ± iqrs	1	0.356 ± 0.0387	0.474 ± 0.0578
		5	0.346 ± 0.0242	0.456 ± 0.0475
		10	$\mathbf{0.346} \pm 0.0268$	$\mathbf{0.447} \pm 0.0529$
SVIGP	Means ± stds	$m = 10$	0.337 ± 0.0284	0.26 ± 0.0272
		$m = 50$	0.334 ± 0.0162	$\mathbf{0.234} \pm 0.022$
		$m = 200$	0.324 ± 0.02	0.237 ± 0.0354
		$m = 455$	$\mathbf{0.324} \pm 0.021$	0.242 ± 0.0402
	Medians ± iqrs	$m = 10$	0.331 ± 0.0175	0.256 ± 0.0479
		$m = 50$	0.332 ± 0.0125	$\mathbf{0.229} \pm 0.0389$
		$m = 200$	$\mathbf{0.322} \pm 0.0272$	0.230 ± 0.0584
		$m = 455$	0.326 ± 0.0286	0.235 ± 0.0713

Table 3.3: The test RMSEs of the RNVGP run on the Boston and SARCOS data sets, stratified by d. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison.

The test RMSEs of the RNVGP stratified by the trace product randomisation parameter d are presented in Table 3.3. We observe for the Boston results that increasing d from 1 to 5 improves the test RMSE, but further increase offers no benefit. By contrast, the SARCOS test RMSE is uniformly reduced by increasing d.

We observe relatively small dispersion within each cell, again suggesting that the inference is quite consistent. However, we observe that the median test RMSE is consistently smaller than the mean RMSE, suggesting a positive skew in the results.

Method	Time (s)	d	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	1	**457** ± 438	**632** ± 627
		5	876 ± 858	1.19e3 ± 1.2e3
		10	1.19e3 ± 1.12e3	1.69e3 ± 1.68e3
	Medians ± iqrs	1	**230** ± 884	**292** ± 1.25e3
		5	463 ± 1.61e3	560 ± 2.4e3
		10	678 ± 1.87e3	864 ± 2.98e3
SVIGP	Means ± stds	$m = 10$	**302** ± 7.86	**296** ± 11.8
		$m = 50$	331 ± 18.4	349 ± 20.4
		$m = 200$	995 ± 101	1.03e3 ± 120
		$m = 455$	4.72e3 ± 237	6.39e3 ± 396
	Medians ± iqrs	$m = 10$	**302** ± 14.0	**295** ± 23.5
		$m = 50$	327 ± 27.6	345 ± 30.9
		$m = 200$	969 ± 142.0	1.03e3 ± 161
		$m = 455$	4.72e3 ± 377	6.37e3 ± 881

Table 3.4: The computational times in seconds of the RNVGP run on the Boston and SARCOS data sets, stratified by d. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each data set for comparison.

In Table 3.4 we observe an increase in computational time with d, although weaker than the anticipated $O(d)$ scaling. This again suggests that this is not the rate-limiting step in the computation. The dispersion in computational times is quite large, although the average statistics are consistently smaller than the SVIGP.

3.3.2.4 RNVGP Results Stratified by b

Table 3.5 presents the RNVGP test RMSEs separated by the batch size b. We see that the Boston test RMSEs are improved by the initial increase in b, but soon asymptote to an optimal value. By contrast, the SARCOS runs continue to improve with larger b: this is perhaps to be expected, as the SARCOS data set has far more data points, meaning that the variance introduced by the batches will be larger. We also see that the dispersion within the cells is smaller for the Boston than SARCOS runs, suggesting a more consistent inference for the smaller data set.

Method	Test RMSE	b	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	1	0.396 ± 0.0346	0.501 ± 0.0607
		10	0.35 ± 0.0187	0.473 ± 0.0679
		50	0.345 ± 0.015	0.462 ± 0.0529
		100	$\mathbf{0.345} \pm 0.0143$	$\mathbf{0.455} \pm 0.057$
	Medians ± iqrs	1	0.395 ± 0.0471	0.484 ± 0.0724
		10	0.346 ± 0.0174	0.457 ± 0.0473
		50	$\mathbf{0.343} \pm 0.0188$	0.451 ± 0.0429
		100	0.344 ± 0.0169	$\mathbf{0.444} \pm 0.0414$
SVIGP	Means ± stds	$m = 10$	0.337 ± 0.0284	0.26 ± 0.0272
		$m = 50$	0.334 ± 0.0162	$\mathbf{0.234} \pm 0.022$
		$m = 200$	0.324 ± 0.02	0.237 ± 0.0354
		$m = 455$	$\mathbf{0.324} \pm 0.021$	0.242 ± 0.0402
	Medians ± iqrs	$m = 10$	0.331 ± 0.0175	0.256 ± 0.0479
		$m = 50$	0.332 ± 0.0125	$\mathbf{0.229} \pm 0.0389$
		$m = 200$	$\mathbf{0.322} \pm 0.0272$	0.23 ± 0.0584
		$m = 455$	0.326 ± 0.0286	0.235 ± 0.0713

Table 3.5: The test RMSEs of the RNVGP run on the Boston and SARCOS data sets, stratified by b. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison.

Regarding the computational times stratified by b presented in Table 3.6, we observe a weak increase with b, suggesting that this is not the computational bottleneck of the algorithm. We also observe large but consistent dispersions within each cell, with the SARCOS data set exhibiting more inconsistency in computational time. We observe that the average RNVGP is smaller than the corresponding SVIGP average computational time.

Method	Time (s)	b	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	1	**749** ± 827	**1.06**e3 ± 1.25e3
		10	818 ± 901	1.09e3 ± 1.24e3
		50	844 ± 870	1.15e3 ± 1.25e3
		100	951 ± 997	1.37e3 ± 1.5e3
	Medians ± iqrs	1	**384** ± 826	**469** ± 1.18e3
		10	437 ± 900	535 ± 1.24e3
		50	486 ± 968	593 ± 1.38e3
		100	550 ± 1.08e3	747 ± 1.55e3
SVIGP	Means ± stds	$m = 10$	**302** ± 7.86	**296** ± 11.8
		$m = 50$	331 ± 18.4	349 ± 20.4
		$m = 200$	995 ± 101	1.03e3 ± 120
		$m = 455$	4.72e3 ± 237	6.39e3 ± 396
	Medians ± iqrs	$m = 10$	**302** ± 14.0	**295** ± 23.5
		$m = 50$	327 ± 27.6	345 ± 30.9
		$m = 200$	969 ± 142	1.03e3 ± 161
		$m = 455$	4.72e3 ± 377	6.37e3 ± 881

Table 3.6: The computational times in seconds of the RNVGP run on the Boston and SARCOS data sets, stratified by b. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison.

3.3.2.5 RNVGP Results Stratified by ϵ_L and ϵ

The test RMSEs for the RNVGP run on the Boston and SARCOS data sets are presented in Table 3.7, stratified by ϵ_L and ϵ. We see that reducing the tolerances of the randomised stopping algorithm improves the predictive performance of the models. The difference in the means and medians of the results again suggests a positive skew in the RMSE distribution: the Boston results exhibit a relatively small dispersion, whereas the SARCOS results have a larger dispersion.

Considering the timing results stratified by ϵ_L and ϵ presented in Table 3.8, we see that decreasing the tolerance of the algorithm moderately increases the computational time of the algorithm, suggesting that this is not a computationally limiting step of the algorithm.

Method	Test RMSE	ϵ_L, ϵ	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	0.1, 0.001	0.365 ± 0.0342	0.495 ± 0.0647
		0.001, $1e-06$	$\mathbf{0.352} \pm 0.0255$	$\mathbf{0.451} \pm 0.0511$
	Medians ± iqrs	0.1, 0.001	0.354 ± 0.0371	0.478 ± 0.0555
		0.001, $1e-06$	$\mathbf{0.345} \pm 0.0234$	$\mathbf{0.441} \pm 0.0406$
SVIGP	Means ± stds	$m = 10$	0.337 ± 0.0284	0.26 ± 0.0272
		$m = 50$	0.334 ± 0.0162	$\mathbf{0.234} \pm 0.022$
		$m = 200$	0.324 ± 0.02	0.237 ± 0.0354
		$m = 455$	$\mathbf{0.324} \pm 0.021$	0.242 ± 0.0402
	Medians ± iqrs	$m = 10$	0.331 ± 0.0175	0.256 ± 0.0479
		$m = 50$	0.332 ± 0.0125	$\mathbf{0.229} \pm 0.0389$
		$m = 200$	$\mathbf{0.322} \pm 0.0272$	0.230 ± 0.0584
		$m = 455$	0.326 ± 0.0286	0.235 ± 0.0713

Table 3.7: The test RMSEs of the RNVGP run on the Boston and SARCOS data sets, stratified by ϵ_L and ϵ. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison.

Method	Time (s)	ϵ_L, ϵ	Data: Boston	Data: SARCOS
RNVGP	Means ± stds	0.1, 0.001	**716** ± 729	**1.05e3** ± 1.15e3
		0.001, $1e-06$	965 ± 1.04e3	1.29e3 ± 1.46e3
	Medians ± iqrs	0.1, 0.001	**403** ± 933	**505** ± 1.36e3
		0.001, $1e-06$	515 ± 1.01e3	610 ± 1.43e3
SVIGP	Means ± stds	$m = 10$	**302** ± 7.86	**296** ± 11.8
		$m = 50$	331 ± 18.4	349 ± 20.4
		$m = 200$	995 ± 101	1.03e3 ± 120
		$m = 455$	4.72e3 ± 237	6.39e3 ± 396
	Medians ± iqrs	$m = 10$	**302** ± 14.0	**295** ± 23.5
		$m = 50$	327 ± 27.6	345 ± 30.9
		$m = 200$	969 ± 142	1.03e3 ± 161
		$m = 455$	4.72e3 ± 377	6.37e3 ± 881

Table 3.8: The computational times in seconds of the RNVGP run on the Boston and SARCOS data sets, stratified by ϵ_L and ϵ. The results are presented as means and medians across runs with standard deviations and interquartile ranges, respectively. We also see the SVIGP results averaged across all runs for each m and data set for comparison.

3.3.2.6 RNVGP Results Stratified by m, b, d, ϵ_L and ϵ

In Appendix B, we present the means and medians of the test RMSEs and computational times, stratified by every experimental variable except for the random seed. These results can be seen in Tables B.1, B.2, B.3, B.4, B.5, B.6, B.7 and B.8. We do not see any trends in the results that were not evident from the singly-stratified results, but it is instructive to consider the uncertainty in each cell, as it is an indication of how robust the inference is when varying

just the random seed. We again see that the inference for the SARCOS data set was much more variable than the Boston data set.

3.3.2.7 Best Performing RNVGP Experiments

In Table 3.9, we see the five best performing experimental conditions for the Boston data set, averaged over random seeds. The test RMSEs and times both exhibit relatively low standard deviations, suggesting that consistent optimisation is being achieved. We see that large values of $m = 455$ and smaller values of $\epsilon_L = 0.001$ and $\epsilon = 1\text{e} - 6$ are generally preferred, but no consistent preference for b and d among the best five runs. Curiously, the run with maximum b and d is not found among the top runs, but neither was any run with small b and d. We also observe an intriguing decrease in the log variational bound with performance for the conditions with the five lowest test RMSE.

m	b	d	ϵ_L, ϵ	Test RMSE	Time (s)	Log Bound
455	100	5	0.001, $1e-06$	$0.327 \pm 5.55\text{e} - 3$	$2.45\text{e}3 \pm 123$	-3.12 ± 0.927
455	50	5	0.001, $1e-06$	$0.328 \pm 5.74\text{e} - 3$	$2.5\text{e}3 \pm 93.1$	-1.73 ± 0.688
455	50	10	0.001, $1e-06$	$0.334 \pm 6.54\text{e} - 3$	$2.97\text{e}3 \pm 145$	-1.92 ± 0.853
455	10	5	0.001, $1e-06$	$0.334 \pm 8.55\text{e} - 3$	$2.3\text{e}3 \pm 86.1$	-0.365 ± 0.118
455	10	10	0.001, $1e-06$	$0.334 \pm 6.97\text{e} - 3$	$3.19\text{e}3 \pm 234$	-0.485 ± 0.469

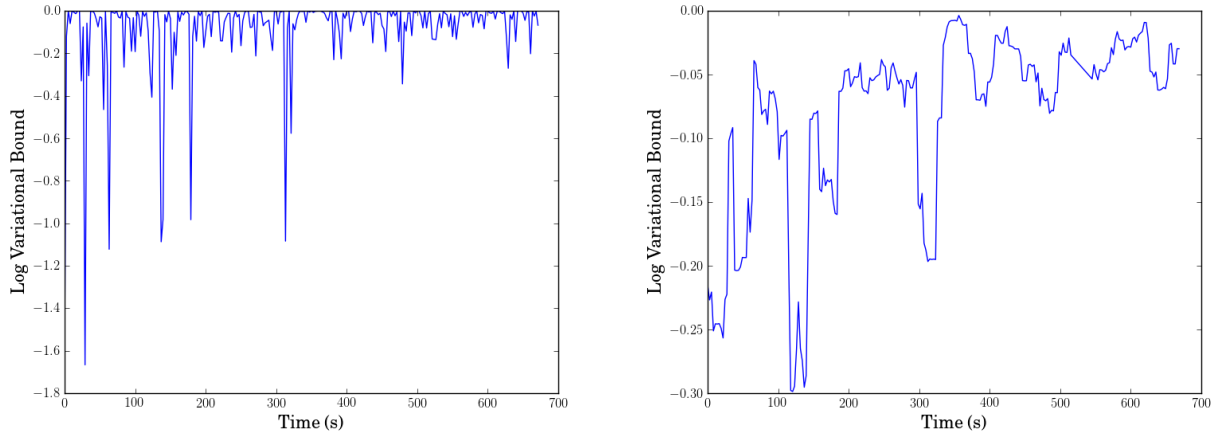
Table 3.9: The best performing experimental conditions for the RNVGP on the Boston data set, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs, times in seconds and final evaluations of the log variational bound.

Similarly, in Table 3.10 we see the five best performing experimental conditions for the SARCOS data set, averaged over random seed. The test RMSEs and times both exhibit higher standard deviations than for the Boston data set, suggesting that the optimisation is less consistent. All of the top five runs have the lowest values of ϵ_L and ϵ and moderate to large values of m, b and d. We see the runs with maximum b and d feature three times, unlike for the Boston experiments. We observe no clear link between the variational bound value and the test RMSE.

m	b	d	ϵ_L, ϵ	Test RMSE	Time (s)	Log Bound
200	100	10	0.001, $1e-06$	0.413 ± 0.0134	$1.15\text{e}3 \pm 30.0$	-8.68 ± 2.3
500	100	10	0.001, $1e-06$	0.413 ± 0.0156	$5.19\text{e}3 \pm 278$	-9.56 ± 3.54
500	10	5	0.001, $1e-06$	0.415 ± 0.0201	$2.96\text{e}3 \pm 127$	-0.698 ± 0.386
500	100	5	0.001, $1e-06$	0.415 ± 0.0281	$3.48\text{e}3 \pm 334$	-9.66 ± 2.96
200	10	10	0.001, $1e-06$	$0.417 \pm 7.5\text{e} - 3$	884 ± 24.8	-0.538 ± 0.158

Table 3.10: The best performing experimental conditions for the RNVGP on the SARCOS data set, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs, times in seconds and final evaluations of the log variational bound.

In Figure 3.2 we see the optimisation of the RNVGP bound with the lowest final test RMSE on the Boston data set. We see, more clearly with the smoothed results, that the optimisation appears to gradually converge to a reasonably stable mode.



(a) Unsmoothed (b) Smoothed

Figure 3.2: The RNVGP run with lowest test RMSE on the Boston data set, with $m = 200$, $d = 10$, $b = 1$, $\epsilon_L = 0.001$, $\epsilon = 1\text{e} - 6$ and random seed equal to 9. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.314 and took a time of 671 seconds to optimise.

The single best performing run of the RNVGP on the SARCOS data set is plotted in Figure 3.3. We see that it has a large value of $m = 500$, and moderate to large values of $d = 5$ and $b = 100$. The trajectory of the optimisation appears to follow a clear, if noisy, convergence to a stable mode.

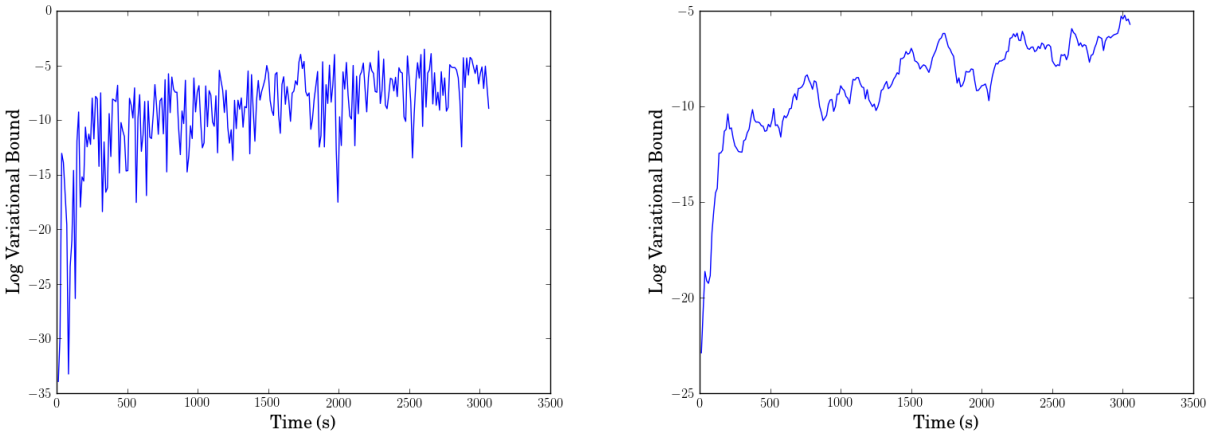
3.3.3 SVIGP Experimental Results

For comparison, here we more thoroughly present the experimental results for the SVIGP on the Boston and SARCOS data sets, given the experimental design explained earlier.

3.3.3.1 Boston Housing Data

		m			
Test RMSEs	b	10	50	200	455
Means ± stds	1	0.368 ± 0.0395	0.346 ± 0.0232	0.327 ± 0.025	0.334 ± 0.0272
	10	0.324 ± 0.0137	0.323 ± 0.014	0.337 ± 0.0163	0.332 ± 0.0123
	50	$0.333 \pm 5.98\text{e} - 3$	$0.335 \pm 5.02\text{e} - 3$	$0.328 \pm 5.18\text{e} - 3$	$0.33 \pm 5.77\text{e} - 3$
	100	$0.322 \pm 4.72\text{e} - 3$	$0.330 \pm 3.52\text{e} - 3$	$0.303 \pm 6.49\text{e} - 3$	$\mathbf{0.299} \pm 4.65\text{e} - 3$
Medians ± iqrs	1	0.361 ± 0.0134	0.345 ± 0.0311	0.321 ± 0.036	0.334 ± 0.0268
	10	0.323 ± 0.0177	0.325 ± 0.0225	0.338 ± 0.0234	0.327 ± 0.0195
	50	0.333 ± 0.0111	$0.337 \pm 6.36\text{e} - 3$	$0.329 \pm 5.75\text{e} - 3$	$0.328 \pm 8.39\text{e} - 3$
	100	$0.323 \pm 6.3\text{e} - 3$	$0.331 \pm 1.67\text{e} - 3$	0.305 ± 0.012	$\mathbf{0.297} \pm 5.64\text{e} - 3$

Table 3.11: The test RMSEs of the SVIGP run on the Boston data set, stratified by b and m. Means across runs are presented with standard deviations, and medians with interquartile ranges.



(a) Unsmoothed (b) Smoothed

Figure 3.3: The RNVGP run with lowest test RMSE on the SARCOS data set, with $m = 500$, $d = 5$, $b = 100$, $\epsilon_L = 0.001$, $\epsilon = 1\text{e} - 6$ and random seed equal to 3. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.364 and took a time of 3,070 seconds to optimise.

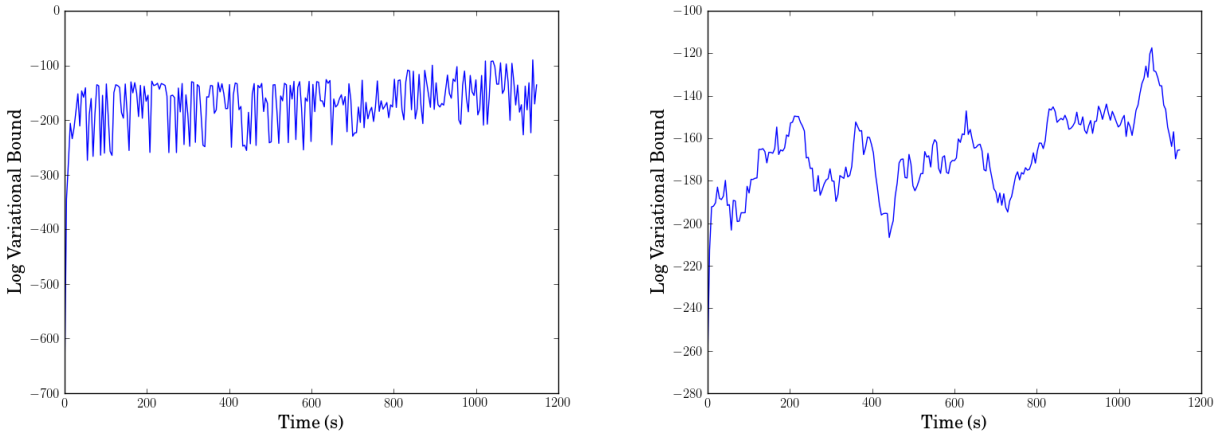
In Figure 3.11 we observe the test RMSEs for the SVIGP on the Boston data set, stratified by b and m. Consequently, the variation within each cell is solely due to differences in random seeds. We can immediately observe very low variation within each cell, suggesting that there is very consistent inference for the SVIGP bound. We also see that there is little difference between the means and medians, suggesting that there is very little skew in the distribution of the results.

Considering the variable b, we see that the increase of this variable generally improves the performance of the methods, although the results for $b = 1$ still optimise reasonably successfully. We observe that the test RMSE generally decreases with m, with some exceptions.

		m			
Time (s)	b	10	50	200	455
Means ± stds	1	**297** ± 5.08	312 ± 1.57	892 ± 2.58	4.52e3 ± 178
	10	301 ± 7.16	318 ± 1.71	919 ± 1.41	4.59e3 ± 162
	50	309 ± 4.2	336 ± 0.989	1.02e3 ± 1.14	4.72e3 ± 16.0
	100	303 ± 9.02	359 ± 0.833	1.15e3 ± 3.0	5.05e3 ± 21.3
Medians ± iqrs	1	**296** ± 6.92	312 ± 2.54	892 ± 3.28	4.47e3 ± 313
	10	299 ± 10.2	318 ± 1.59	920 ± 1.42	4.57e3 ± 284
	50	309 ± 4.32	336 ± 1.66	1.02e3 ± 0.996	4.72e3 ± 27.2
	100	307 ± 15.1	359 ± 1.05	1.15e3 ± 5.19	5.05e3 ± 18.5

Table 3.12: The computational times in seconds of the SVIGP run on the Boston data set, stratified by b and m. Means across runs are presented with standard deviations, and medians with interquartile ranges.

The computational times for the SVIGP run on the Boston data set are presented in Figure 3.12, stratified by b and m. We again notice small variation within cells brought about by



(a) Unsmoothed (b) Smoothed

Figure 3.4: The SVIGP run with lowest test RMSE on the Boston data set, with $m = 200$, $b = 100$ and random seed equal to 6. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.292 and took a time of 1,150 seconds to optimise.

differences in random seeds. We also observe clear strong positive scaling with m, and weak positive scaling with b, suggesting that it is the $O(m^3)$ computations which are the rate-limiting operation.

m	b	Test RMSE	Time (s)	Log Bound
455	100	$0.299 \pm 4.65\text{e}-3$	$5.05\text{e}3 \pm 21.3$	$-153. \pm 30.3$
200	100	$0.303 \pm 6.49\text{e}-3$	$1.15\text{e}3 \pm 3.0$	$-167. \pm 31.0$
10	100	$0.322 \pm 4.72\text{e}-3$	303.0 ± 9.02	$-263. \pm 43.7$
50	10	0.323 ± 0.014	318.0 ± 1.71	$-294. \pm 442.0$
10	10	0.324 ± 0.0137	301.0 ± 7.16	$-309. \pm 178.0$

Table 3.13: The best performing experimental conditions for the SVIGP on the Boston data set, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs times in seconds, and final evaluations of the log variational bound.

The most successful experimental conditions averaged over random seed for the Boston data set are presented in Table 3.13. We see that the best two performing conditions feature the largest value of b and the two largest values of m, although strangely the next best successful conditions include much smaller and computationally cheaper values of m and b.

The single best run of the SVIGP on the Boston data set is seen in Figure 3.4, with the variational bound plotted over the course of optimisation, in both a smoothed and unsmoothed form. We see that the value of the bound quickly converges to a reasonably stable mode.

3.3.4 SARCOS Data

Test RMSEs	b	m = 10	50	200	500
Means ± stds	1	0.291 ± 0.0131	$0.261 \pm 6.33\text{e}-3$	0.286 ± 0.0162	0.297 ± 0.0102
	10	0.278 ± 0.0134	0.248 ± 0.0112	$0.251 \pm 8.42\text{e}-3$	0.26 ± 0.0109
	50	$0.24 \pm 8.06\text{e}-3$	$0.217 \pm 3.01\text{e}-3$	$0.211 \pm 6.36\text{e}-3$	$0.212 \pm 3.68\text{e}-3$
	100	$0.231 \pm 5.4\text{e}-3$	$0.211 \pm 5.83\text{e}-3$	$0.199 \pm 3.36\text{e}-3$	$\mathbf{0.197} \pm 2.01\text{e}-3$
Medians ± iqrs	1	0.287 ± 0.0184	$0.261 \pm 7.81\text{e}-3$	0.291 ± 0.0317	0.299 ± 0.0152
	10	$0.278 \pm 9.6\text{e}-3$	$0.246 \pm 9.48\text{e}-3$	0.254 ± 0.012	0.257 ± 0.0188
	50	$0.236 \pm 8.15\text{e}-3$	$0.216 \pm 3.39\text{e}-3$	0.211 ± 0.0106	$0.210 \pm 2.18\text{e}-3$
	100	$0.231 \pm 2.64\text{e}-3$	$0.210 \pm 8.54\text{e}-3$	$0.200 \pm 4.48\text{e}-3$	$\mathbf{0.198} \pm 2.7\text{e}-3$

Table 3.14: The test RMSEs of the SVIGP run on the SARCOS data set, stratified by b and m. Means across runs are presented with standard deviations, and medians with interquartile ranges.

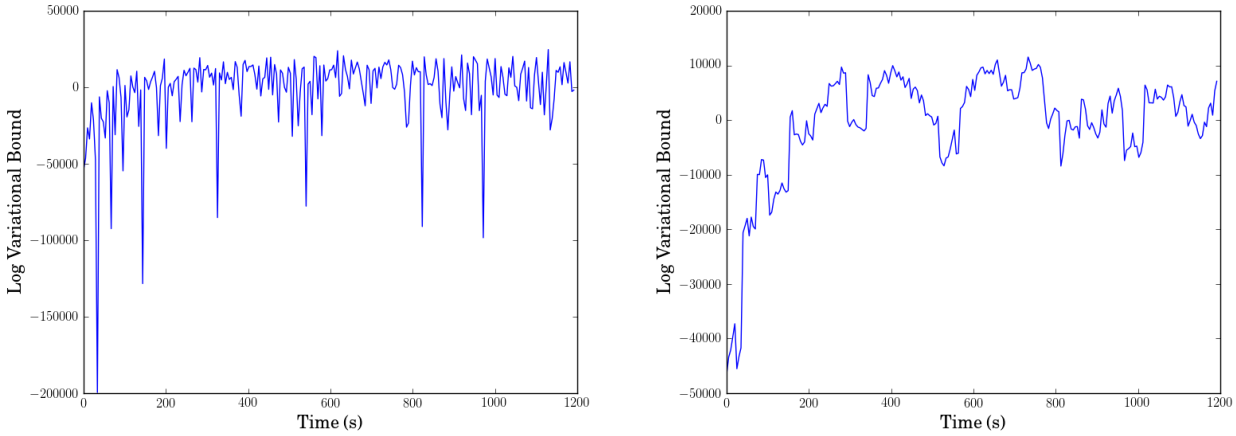
In Table 3.14 we see the RMSE on the test data of the SVIGP run on the SARCOS data set, stratified by b and m. We again observe very small variation within cells, and small difference between the mean and median statistics.

Increasing b has a much more positive effect on the test RMSE than with the Boston data set: this might be expected, as the Boston data set is far smaller, so the data will be resampled more frequently and the batch-wise inference consequently contributes less noise to the gradients. We again observe a general increase in performance with m, most markedly for larger b. It appears that the larger variational distributions need more information to optimise well.

Time (s)	b	m = 10	50	200	500
Means ± stds	1	$\mathbf{282} \pm 1.2$	327 ± 9.21	922 ± 41.9	$6.09\text{e}3 \pm 226$
	10	292 ± 6.29	340 ± 11.4	927 ± 1.83	$6.04\text{e}3 \pm 225$
	50	300 ± 8.69	350 ± 8.22	$1.05\text{e}3 \pm 0.871$	$6.58\text{e}3 \pm 267$
	100	310 ± 4.83	377 ± 5.76	$1.2\text{e}3 \pm 1.75$	$6.84\text{e}3 \pm 17.9$
Medians ± iqrs	1	$\mathbf{282} \pm 2.03$	331 ± 13.0	901 ± 7.76	$5.92\text{e}3 \pm 411$
	10	293 ± 7.28	340 ± 7.37	927 ± 2.4	$5.94\text{e}3 \pm 6.88$
	50	302 ± 16.4	345 ± 14.7	$1.05\text{e}3 \pm 0.498$	$6.38\text{e}3 \pm 417$
	100	310 ± 6.24	375 ± 1.47	$1.2\text{e}3 \pm 2.06$	$6.84\text{e}3 \pm 29.7$

Table 3.15: The computational times in seconds of the SVIGP run on the SARCOS data set, stratified by b and m. Means across runs are presented with standard deviations, and medians with interquartile ranges.

In Table 3.15 we see the computational times for the SVIGP run on the SARCOS data, stratified by b and m. There remains little variation within cells, and little difference between the means and medians. There is strong positive scaling with m and weak scaling with b, suggesting that the $O(m^3)$ computations are the rate limiting step.



(a) Unsmoothed

(b) Smoothed

Figure 3.5: The SVIGP run with lowest test RMSE on the SARCOS data set, with $m = 200$, $b = 100$ and random seed equal to 8. The plot on the left shows the variational bound being optimised, the plot on the right shows the same bound smoothed over a window of ten iterations. The predictions had a test RMSE of 0.193 and took a time of 1,200 seconds to optimise.

m	b	Test RMSE	Time (s)	Log Bound
500	100	$0.197 \pm 2.01\text{e}-3$	$6.84\text{e}3 \pm 17.9$	$3.00\text{e}3 \pm 2.47\text{e}4$
200	100	$0.199 \pm 2.74\text{e}-3$	$1.21\text{e}3 \pm 34.6$	$-1.01\text{e}4 \pm 8.58\text{e}3$
200	50	$0.211 \pm 6.36\text{e}-3$	$1.05\text{e}3 \pm 0.871$	$-3.19\text{e}3 \pm 1.91\text{e}4$
50	100	$0.211 \pm 5.83\text{e}-3$	377.0 ± 5.76	$-5.09\text{e}3 \pm 2.85\text{e}4$
500	50	$0.212 \pm 3.68\text{e}-3$	$6.58\text{e}3 \pm 267.0$	$-3.62\text{e}3 \pm 1.26\text{e}4$

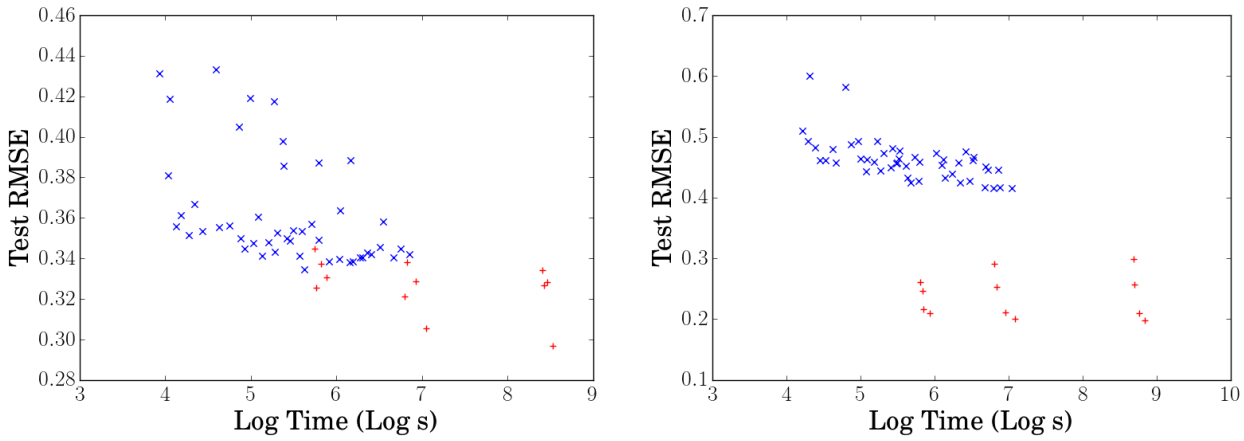
Table 3.16: The best performing experimental conditions for the SVIGP on the SARCOS data set, averaged over random seeds. The results are presented as means and standard deviations of test RMSEs times in seconds and final evaluations of the log variational bound.

Examining the best performing experimental conditions for the SARCOS data set in Table 3.16, it is clear that the larger values of m and b are the most successful. The optimisation of the bound for the single best run is presented in smoothed and unsmoothed form in Figure 3.5: we see that the optimisation converges to a mode fairly quickly and stably.

3.4 Comparison of RNVGP and SVIGP Results

We can see from a comparison of the RNVGP results and the SVIGP results that the new method exhibits mixed performance in comparison to the SVIGP. The medians taken across random seed of the test RMSE and log computational time are presented as scatter plots for the Boston and SARCOS data sets in Figure 3.6.

For the Boston data set, the RNVGP runs perform moderately less accurately than the SVIGP method, with a corresponding average decrease in computation time. For the SARCOS data set, the difference in test RMSE is much more stark, without a correspondingly large trade-off in computational time. It appears that the inference for the RNVGP on the SARCOS data set could be heavily suboptimal, as the difference in test RMSE with the SVIGP is very



(a) Boston (b) SARCOS

Figure 3.6: Scatter plots of test RMSE medians and log time medians taken across random seeds. SVIGP results are presented as plus signs, and RNVGP results as crosses.

pronounced and the large dispersion between different SARCOS experiments suggests that the inference is inconsistent.

It might be expected that test RMSE will be worse for the RNVGP, as the inference was performed with noisier gradients and a more restrictive variational bound. It is not clear that the theoretical $O(m^2)$ computational cost is providing a sufficient timing trade-off to make the RNVGP attractive in comparison to the $O(m^3)$ SVIGP. The difference in test RMSE for the experiments performed on SARCOS suggests that for data sets with very large n or p, then the extra gradient variance introduced by the RNVGP makes the inference intractable. For the experiments performed on the smaller Boston data set, it is possible that the reduced accuracy of the RNVGP can be conceded in favour of its more attractive computational time.

3.4.1 Hyperparameter Comparison

Here we compare the ARD length scales of the RNVGP and SVIGP runs with lowest test RMSE on the SARCOS data set. The 21 optimised parameters are presented in Table 3.17.

p	RNVGP l_p	SVIGP l_p	p	RNVGP l_p	SVIGP l_p	p	RNVGP l_p	SVIGP l_p
1	3.802	15.185	8	21.455	13.987	15	4.185	10.456
2	12.525	17.157	9	8.545	16.624	16	12.283	29.498
3	9.381	27.174	10	7.621	31.061	17	11.799	22.806
4	3.141	13.705	11	9.893	17.191	18	8.772	11.650
5	8.757	6.872	12	6.426	28.415	19	8.889	26.885
6	9.542	21.592	13	6.447	25.295	20	8.662	26.582
7	4.792	11.588	14	4.262	19.808	21	4.622	16.737

Table 3.17: The optimised ARD length scales of each dimension for the RNVGP and SVIGP runs with lowest test RMSE on the SARCOS data set, i.e. the SVIGP with $m = 200$, $b = 100$ and random seed equal to 8, and the RNVGP run with $m = 500$, $d = 5$, $b = 100$, $\epsilon_L = 0.001$, $\epsilon = 1\text{e} - 6$ and random seed equal to 3.

We see in Table 3.17 that the SVIGP length scales are on average larger than those found by the RNVGP. We do not observe a strong correlation between the length scales found by each method: it appears that they are not consistently selecting the same dimensions as powerful predictors. It is plausible that the modified variational representation used by the RNVGP demands different hyperparameters, but is is also possible that the hyperparameters are not fully optimised for the more challenging RNVGP inference. However, it is worth asserting that it is difficult to interpret ARD lengthscales in high dimensions, and differing values of kernel hyperparameters do not necessarily imply that the models are at odds with one another.

3.5 Conclusions and Future Work

It can be seen that the new RNVGP is not clearly more successful than the standard SVIGP. It is not clear whether the difference in performance is due to the more challenging inference or the more heavily restricted variational bound.

Further work would involve systematically exploring these potential problems. It is clear from the results that larger values of b and smaller values of ϵ_L and ϵ may yet improve performance, and warrant further exploration. It is also possible that the optimisation updates and schedule used could be improved: exploring the use of Adam, AdaDelta or other adaptive methods for the RNVGP could prove beneficial (Ruder, 2016).

It would also be possible to explore different representations of the variational bound, although it would need to be within a family of distributions that permits the use of the stochastic computational methods that we are interested in exploiting.

It is possible that the new representation of the covariance is making the hyperparameter inference challenging: the instability in the length scale and variance optimisation is a fairly common problem with variational methods. Similarly, we have not explored the properties of the predictive variance in this chapter, although it would be important future work to ensure that the $O(m)$ variational representation produces predictive distributions with well-calibrated uncertainty.

The use of preconditioning and fast CMVPs is also explored in (Filippone and Engler, 2015), and they "observe that they yield little gain in computational speed compared to the standard CG algorithm." However, it is possible that the differing properties of the variational inference will cause these methods to have greater power in this context. It would be possible to investigate the relevance of these methods for the RNVGP.

Chapter 4

Bayesian Co-Training with Differential Privacy

The problem of model combination with Gaussian Processes is explored, specifically in a context in which privacy is valued. Analogies to co-training are made, with information exchanged between models by sharing predictions on unlabelled data, both with and without enforcement of differential privacy of the data labels. Multiple model combination methods are introduced and experimental results are presented and analysed. This work was a collaboration between Chris Holmes and myself. All of the code and all of this chapter were written by Owen Thomas.

In this chapter, we pursue the project of successful co-training between Gaussian Processes under the constraint of differential privacy. In Section 4.1 we discuss Bayesian model combination and the co-training algorithm, leading to Section 4.2, in which we establish a framework for co-training between Gaussian Processes without sharing data values. Section 4.3 follows with some results using the framework on simulated data. In Section 4.4 we introduce concepts of differential privacy, recalling the mechanisms discusses in Section 1.4, and also include some similar experiments on simulated data. Section 4.5 consists of experiments and results run on the real CCPP data set. Section 4.6 concludes and discusses future work.

4.1 Model Combination and Co-Training

4.1.1 Model Combination

The coherent treatment of probability within Bayesian inference lends it very naturally to problems associated with model comparison. The posterior probabilities $p(\mathcal{M}_i|\mathcal{D})$ of models $\mathcal{M}_i$ given data $\mathcal{D}$ can be compared, and Bayes factors can be used to judge the relative merits of the models (Bernardo and Smith, 2001).

Posterior probabilities can also be used for the task of model combination, in which we are interested in combining the predictions of several models to produce a joint distribution. The most immediate solution within a Bayesian framework is that of Bayesian Model Averaging (BMA), in which the predictive distributions of each model are weighted proportionally to

the posterior probabilities of each model (Clyde and Iversen, 2013). This has the intuitively sensible property of giving more weight to models which have greater posterior probability, and as such might be expected to exhibit greater predictive power. Formally, if we are predicting the response variable $\mathbf{y}^*$ given test covariates $\mathbf{X}^*$, the following distribution is used:

$$p(\mathbf{y}^*|\mathbf{X}^*, \mathcal{D}) = \sum_i p(\mathbf{y}^*|\mathbf{X}^*, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D}). \tag{4.1}$$

Bayesian Model Averaging can be shown to be optimal when the true model is known to fall within the joint prior of the models $\mathcal{M}$: this is known as the M-closed framework. There are two other possible situations: the M-complete framework, in which the true model is known but not included among the space of considered models $\mathcal{M}$ for practical reasons, and the M-open framework, in which the true model is unknown, and we are compelled to use those in $\mathcal{M}$ instead.

In the latter two frameworks, BMA can give poor results if all of the models within $\mathcal{M}$ have very low likelihood (Monteith et al., 2011). It is known that BMA will select the model that is asymptotically closest in Kullback-Leibler distance to the true model, rather than the combination of models that is closest (Clyde and Iversen, 2013). As such, BMA can allow one model to massively dominate at the expensive of the others: in a situation where all the models have low likelihood, it is often suboptimal to select one model from a pool of poor models: it would be preferable to combine the predictions in a more sophisticated way.

One such solution comes in the form of Bayesian Model Combination (BMC), in which the posterior probabilities of the models are replaced by learned weights w_i (Monteith et al., 2011; Clyde and Iversen, 2013). This can be interpreted as extending the prior to the class of linear combinations of models, which should not exhibit BMA's behaviour of selecting one poor model from many poor models. It is also possible to impose constraints on the values of the weights: while it can be desirable to constrain the weights to be non-negative and sum to one, it is worth stressing that they do not have an interpretation as posterior probabilities of the models (Clyde and Iversen, 2013), as they have not been introduced within a probabilistic framework. Formally:

$$\mathbb{E}(\mathbf{y}^*|\mathbf{X}^*, \mathcal{D}) = \sum_i w_i \mathbb{E}(\mathbf{y}^*|\mathbf{X}^*, \mathcal{M}_i, \mathcal{D}) \tag{4.2}$$

The weights w_i can be learned using a loss function on the data, possibly including desired regularisation terms.

The approach of learning weights associated with linear combinations of models has parallels in the Machine Learning literature, in which "stacking" takes a very similar approach to combining predictions from various models (Wolpert, 1992). There are other approaches commonly gathered under the title of "ensemble methods", including bootstrap aggregating or "bagging", in which models vote for the most likely data label (Breiman, 1996), and "boosting", in which models are iteratively trained on the damped residuals of preceding models (Schapire, 2003).

4.1.2 Co-Training

Model combination methods can be informed by *a priori* information about the relevant models, or the data on which they have been trained. "Co-training" is one such method that combines assumptions about the structure of the component models to give a principled combination method.

The method considers several "views", models built using different sets of covariates, in order to perform supervised learning (Xu et al., 2013). The key assumption is that of conditional independence between the views' covariates, given the response variable, i.e. $p(\mathbf{X}_i, \mathbf{X}_j|\mathbf{y}) = p(\mathbf{X}_i|\mathbf{y})p(\mathbf{X}_j|\mathbf{y})$. This brings about a factorisation of the likelihood between views, enabling independent inference for each model. Consequently, this method is able to combine models with very different structures, avoiding the need to build joint models in situations where that would be challenging. Similar conditional independence assumptions are also used in the development of Bayesian Classifier Combination methods (Ghahramani and Kim, 2003; Simpson et al., 2011).

There are also often strong computational advantages to being able to build models over subsets of data and recombine afterwards, potentially both from a distributed computing perspective, and to reduce the number of necessary operations for models with superlinear algorithmic complexity.

It is also common to combine multi-view methods with large amounts of unlabelled data in a semi-supervised manner (Blum and Mitchell, 1998). The different views are able to communicate by comparing predictions on the unlabelled data: constraining the models to have similar predictions on unlabelled data can improve joint performance, especially if differences in predictive uncertainty are successfully incorporated. This can be particularly effective if each of the views is an expert in different regions of covariate space, such that the shared predictions will be mismatched in confidence.

Co-training has been pursued with various degrees of rigour in the literature: some initial efforts pursued greedy labelling of unlabelled data and point estimates for model parameters. Later methods have attempted a successful quantification of uncertainty over model parameters and unknown data values: this can be done either by Gibbs sampling, or with an Expectation-Maximisation algorithm, treating the unknown data labels as latent variables and learning point estimates for the model parameters (Nigam and Ghani, 2000).

4.1.3 Bayesian Co-Training with Gaussian Processes

In "Bayesian Co-Training", Yu et al successfully pursue a combination of the co-training framework with Gaussian processes (Yu et al., 2011).

Yu et al define Gaussian latent variables $\mathbf{f}_i$ over each view separately with covariances $\mathbf{K}_i$ which are combined into a consensus function $\mathbf{f}_c$. Conditional independence assumptions are imposed between the latent functions given the response variables, i.e. $p(\mathbf{f}_i, \mathbf{f}_j|\mathbf{y}) = p(\mathbf{f}_i|\mathbf{y})p(\mathbf{f}_j|\mathbf{y})$. Potential functions $\Psi(\mathbf{f}_i, \mathbf{f}_c)$ and $\Psi(\mathbf{f}_c, \mathbf{y})$ are then used to impose agreement between each of

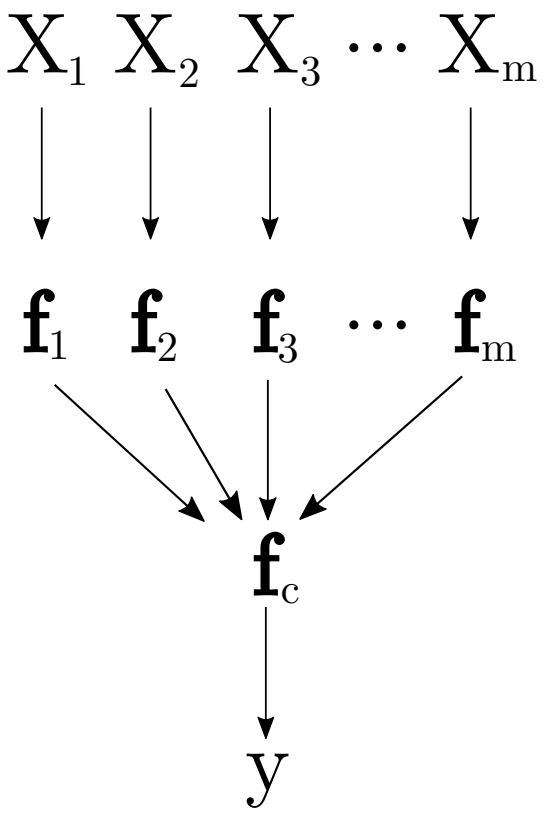


Figure 4.1: A graphical model demonstrating the relationship between the covariates $\mathbf{X}_i$ and latent functions $\mathbf{f}_i$ of each of the m views, the joint consensus function $\mathbf{f}_c$ and the observed response variable $\mathbf{y}$.

the latent variables $\mathbf{f}_i$ and consensus function $\mathbf{f}_c$, and also the consensus function $\mathbf{f}_c$ and the response variable data $\mathbf{y}$. A graphical model of the situation is presented in Figure 4.1.

In the context of regression, an exponentiated quadratic form for the potential function is used, i.e. $\Psi(f_i, f_j) = \exp(-\frac{1}{2\sigma^2}||f_i - f_j||^2)$. Consequently, for some problem-specific potential function Ψ, the joint probability is defined as:

$$p(\mathbf{y}, \mathbf{f}_c, \mathbf{f}_1, ..., \mathbf{f}_m) = \frac{1}{Z}\Psi(\mathbf{y}, \mathbf{f}_c)\prod_i \Psi(\mathbf{f}_i, \mathbf{f}_c) \tag{4.3}$$

Following from this result, it is possibly to marginalise out the individual views' latent variables $\mathbf{f}_i$, resulting in a model with a single model with a single "co-training kernel" over $\mathbf{f}_c$. The kernel offers a way of running a single Gaussian process model with a specific kernel, while implicitly running GPs over each view, and constraining them to agree on unlabelled data points. The kernel is defined as follows:

$$\mathbf{K}_c = \Big[\sum_i (\mathbf{K}_i + \sigma_i^2 I)^{-1}\Big]^{-1} \tag{4.4}$$

Each kernel component $\mathbf{K}_i$ consists of the covariance between all of the covariates visible to view i, including any unlabelled and test data. Covariance components for data points not visible to view i are left as zeros in $\mathbf{K}_i$. The covariance matrix associated with each view

has diagonal noise added and is then inverted to create a precision matrix for view i given by $(\mathbf{K}_i + \sigma_i^2 I)^{-1}$. These are all then added to create a joint precision matrix over all views $\sum_i (\mathbf{K}_i + \sigma_i^2 I)^{-1}$. Inverting the joint precision matrix then produces a joint covariance across views $\mathbf{K}_c$. When evaluating the marginal likelihood for training purposes, the submatrix of $\mathbf{K}_c$ associated with the observed training data labels can be taken and used in the computation of the marginal likelihood or predictive distributions.

If the data points associated with each view do not overlap, then the kernel can be built by building a block-diagonal matrix from the precision matrices associated with each view. The block matrix structure follows from the conditional independence assumption: zeroes in precision matrices suggest the conditional independence of normally distributed variables.

The co-training kernel also provides an analytical solution to the problem of imputing the unknown data labels, as the potential functions between latent functions constrain them to be similar on all the observed covariates, even in the absence of observed response variables.

The kernel is unusual for its transductive property, in which all covariates (including test covariates) are included in the kernel, even at training stages. Such a model will have poor generalisation power to unobserved data, and transductive methods are not universally embraced by the data science community. However, the methods are still valid, as the test response variables are not used during training, and it is their distribution that are being targeted.

Including the test covariates at training means that the computation cost scales cubically with the total number of data: if the data points associated with each model overlap, then this will be more expensive than a standard GP. There is also an inverse of the sum of precision matrices from all of the views, so the operations do not distribute across views. For some data configurations, it is possible to make judicious use of the Woodbury matrix identity to reduce computational costs, but this is not always possible.

4.2 Combining with Predictions

4.2.1 Private Co-Training

The issue of privacy has been of increasing interest to the statistical community recently: problems associated with computing using sensitive data on shared computational resources force statisticians to reconsider traditional methods in the interest of protecting private information.

Several approaches have been proposed to address the problem of privacy-preserving statistics:

- It is possible to pursue statistical results after adding artificial noise to the data (Duchi et al., 2013).
- It is possible to train the models normally but add noise to the results that have been derived (Smith et al., 2016).
- It is possible to train certain models in an encrypted form using methods from homomorphic encryption (Aslett et al., 2015).

A question that is pursued here asks if it is possible for two Gaussian processes to jointly learn from one another via predictions, without compromising the privacy of their data. The approach is close to the idea of co-training, in that we have distinct views of the data which are only communicating using predictions.

We introduce the situation that we consider, illustrated in Figure 4.2: two statisticians are performing supervised learning, predicting a response variable $\mathbf{y}$. They each have realisations of the response variable $\mathbf{y}$, not necessarily corresponding to the same data points. However, they have different views of the data, having access to their owns sets of covariates $\mathbf{X}_1$ and $\mathbf{X}_2$. They each have a private set of labelled data, which they can use to build a private supervised model, but which they are not willing to share with one another. The interpretation within this work is that the agents cannot explicitly share their data labels or the covariance of the latent variable evaluated at the private labelled data.

There is, however, a large amount of unlabelled data, on which they are happy to share their predictions of the response variable. If all of the unlabelled data have covariates available from each view, and the statisticians can agree on indices by which to refer to the unlabelled data, then they will be able to request and communicate their predictions. The method of selecting data points to query the other statistician can vary: here we simply have each GP predict at all of the unlabelled data, select the prediction with highest variance, then query the other statistician for their prediction at this point. Such a scheme based on selecting the highest variance point is known as uncertainty sampling.

The work of Yu et al has established a principled way of performing co-training with Gaussian Processes. However, the computations involved do not respect the privacy considerations that we would like to introduce. Firstly, the computations involve concatenating the response variable from both views into one vector, which would reveal the private response variable to whoever performs the global computation. Secondly, the computation of the "co-training kernel" uses all of the views' covariances over their training covariates, which may violate the privacy concerns involved. As such, the methods used in Yu et al will not fulfil the privacy criteria that we are interested in maintaining.

Here we explore how to achieve successful co-training with Gaussian Processes within a privacy-preserving framework. It is not clear how best to integrate information from one model's prediction into another: it is likely that there will be a trade-off between sharing useful information and compromising personal privacy. It is possible that sharing point estimates of the response variable at the predictive points will be sufficient for successful co-training, although it is likely that including the marginal variances or joint covariance between the predictions will help, albeit with the greater possibility of revealing more of the structure of the private data.

It is also interesting to consider this work from a computational perspective. The division of data for privacy reasons has analogies with work done which divides data for computational reasons. A substantial amount of the work considering scalability of statistical methods and "Big Data" divide the data into subsets, perform computation on each separately, then combine the results in a principled fashion (Tresp, 2000; Minsker et al., 2014). It is interesting to

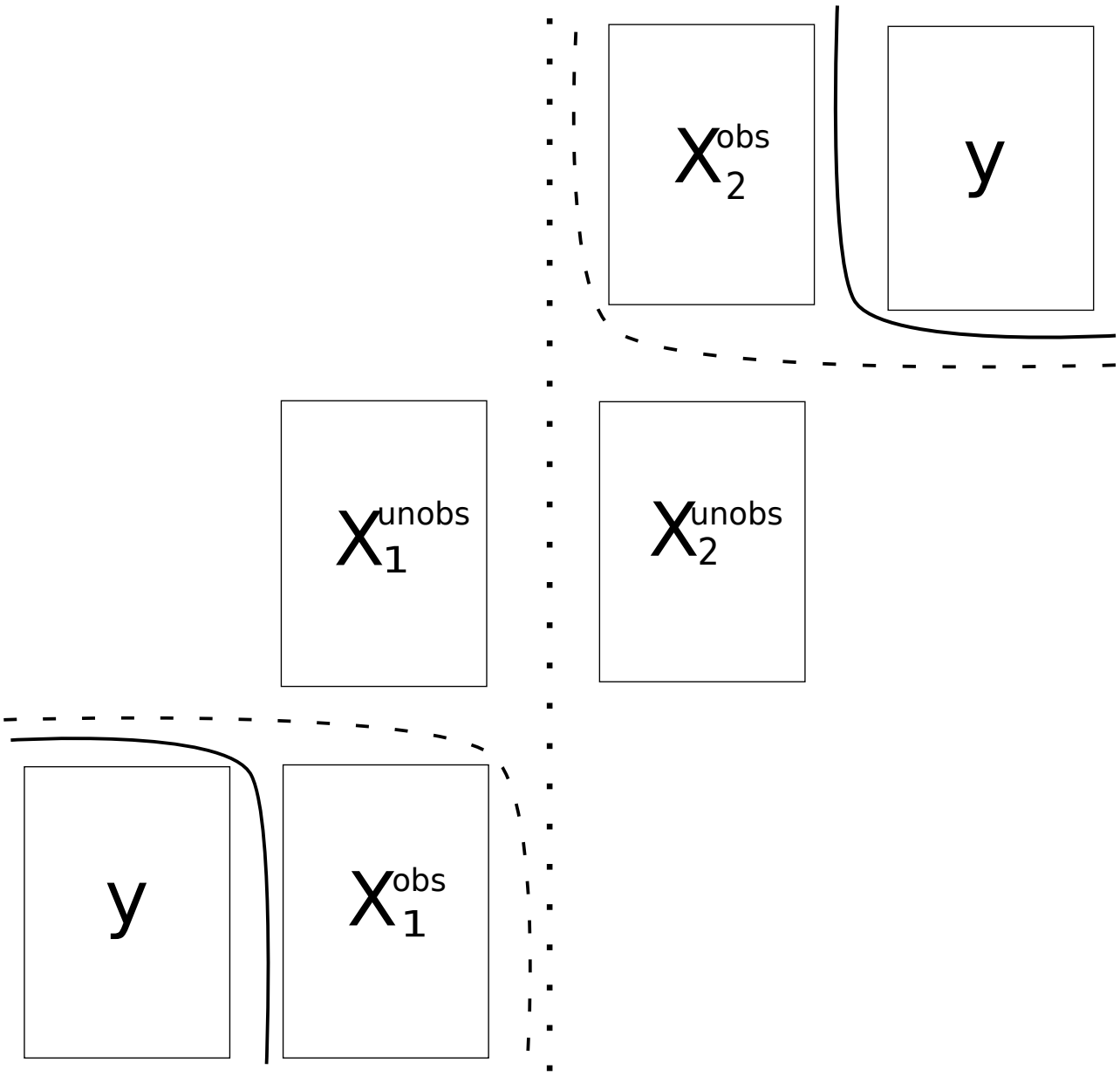


Figure 4.2: A diagram of the experimental setup considered. The three boundaries represent the three degrees of privacy respected: the private response variables $\mathbf{y}$ behind the solid line can be protected by guarantees of differential privacy, their associated covariates $\mathbf{X}^{obs}$ behind the dashed line will not have their values or their joint covariance shared, and the unlabelled covariates $\mathbf{X}^{unobs}$ behind the dotted line will not have their values shared.

reconsider these methods in a privacy-motivated context, since they share structural features with elements of that literature.

4.2.2 Different Combination Approaches

There are several potential methods for combining the predictions of one model with the private observations of another. Ideally, the result would be a generative model that would be able to predict at an arbitrary data point without further communication between statisticians, although this may need to be relaxed to guarantee the coherence of the model.

Three different approaches are outlined here: the "Blocky" approach, the "Noisy" approach, and the "Hybrid" approach.

4.2.2.1 "Blocky" Combination

A method that unambiguously respects the conditional independence of the variables would be one in which the private covariance and the queried covariance are combined in a block-wise fashion to create a joint, blocky covariance matrix for the observed response variables and the queried labels. This should also provide substantial computational advantages, as the blocks can be inverted separately.

Formally, from the perspective of statistician 1:

$$\mathbf{y}_{aug} = (\mathbf{y}_1, \mu(\mathbf{y}_2^{pred})), \tag{4.5}$$

$$\mathbf{K}_{aug} = \left[\begin{array}{c|c} \mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{train}, \theta_1) & 0_{ntrain,nquery} \\ \hline 0_{nquery,ntrain} & \mathbf{K}_2(\mathbf{X}_2^{query}, \mathbf{X}_2^{query}, \theta_1) \end{array}\right] + \left[\begin{array}{c|c} 0_{ntrain,ntrain} & 0_{ntrain,nquery} \\ \hline 0_{nquery,ntrain} & \mathcal{V}_2^{pred} \end{array}\right], \tag{4.6}$$

where $\mu(\mathbf{y}_2^{pred})$ is the reported predictive mean of the second expert on the queried unlabelled data, and $\mathcal{V}_2^{pred}$ is a representation of the uncertainty of the prediction of the second expert.

The "Blocky" kernel consists of a block diagonal matrix, with the blocks consisting of the covariance matrix $\mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{train}, \theta_1)$ associated with the private labelled data of statistician 1 and the reported covariance matrix $\mathbf{K}_2(\mathbf{X}_2^{query}, \mathbf{X}_2^{query}, \theta_1)$ from statistician 2 at the queried data points, including any predictive uncertainty $\mathcal{V}_2^{pred}$.

The question remains how to perform prediction with the "Blocky" model. If the covariance function from only one of the experts is used to evaluate $\mathbf{k}_{\mathbf{ff}^*}$, then the predictive distributions' covariance may not be informative, as the use of kernel between training data and test data will be inconsistent. Using a different kernel during training and testing stages is not necessarily an invalid procedure, but we might expect it to be less successful than a standard method. Conversely, if the block-specific covariance is used, then the model cannot be used to predict at arbitrary test points without reconsulting the other agents for further evaluation of their kernel.

Here we pursue the former path: it is a necessary criterion of the augmented models that they can be used as predictive models for arbitrary test points, and the single covariance for the

test data points is the only one available. The experimental results will hopefully demonstrate whether opting for generalisation over consistency is optimal for this situation.

4.2.2.2 "Noisy" Combination

An intuitively sensible solution to the problem would be to treat the shared prediction as a data point, with a value given by that of the mean of the predictive distribution, and possibly with added heteroscedastic uncertainty associated with the covariance of the distribution. The covariance of the latent variable between all of the data can be evaluated using the kernel function of one of the experts.

By using the same kernel to evaluate the covariance between the training and test data, we should be able to predict on arbitrary data points with the augmented model, while also maintaining positive-definiteness of the predictive distributions.

Formally, from the perspective of statistician 1, with $\mu(\mathbf{y}_2^{pred})$ and $\mathcal{V}_2^{pred}$ defined as earlier:

$$\mathbf{y}_{aug} = (\mathbf{y}_1, \mu(\mathbf{y}_2^{pred})), \tag{4.7}$$

$$\mathbf{K}_{aug} = \left[\begin{array}{c|c} \mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{train}, \theta_1) & \mathbf{K}_1(\mathbf{X}_1^{query}, \mathbf{X}_1^{train}, \theta_1) \\ \hline \mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{query}, \theta_1) & \mathbf{K}_1(\mathbf{X}_1^{query}, \mathbf{X}_1^{query}, \theta_1) \end{array}\right] + \left[\begin{array}{c|c} 0_{ntrain,ntrain} & 0_{ntrain,nquery} \\ \hline 0_{nquery,ntrain} & \mathcal{V}_2^{pred} \end{array}\right]. \tag{4.8}$$

The "Noisy" kernel includes all of the covariance information available from view 1, while including the predictive uncertainty of statistician 2 in the form of $\mathcal{V}_2^{pred}$. This combination method does not include any covariance information from statistician 2 except via the predictive uncertainty $\mathcal{V}_2^{pred}$ which is treated as heteroscedastic noise from the perspective of statistician 1. It is not clear that this method respects the conditional independence assumptions associated with co-training: there is no sum over precisions that is present in the standard co-training kernel.

4.2.2.3 "Hybrid" Combination

It is also of interest to pursue a method with some fidelity to the method from Yu et al (Yu et al., 2011), albeit modified to respect the privacy constraints. A similar transductive kernel was built, incorporating all of the covariance information available from the perspective of one of the agents, using the subset of inverse of sum over inverses method. All that differs from the Yu et al method is the omission of the covariance over the private training data of the other expert, who is represented by their predictions and covariance evaluations on the unlabelled data: these are treated as data points with added uncertainty to the covariance matrix.

Formally, again with $\mu(\mathbf{y}_2^{pred})$ and $\mathcal{V}_2^{pred}$ defined as earlier:

$$\mathbf{y}_{aug} = (\mathbf{y}_1, \mu(\mathbf{y}_2^{pred})), \tag{4.9}$$

$$\mathbf{K}_{aug}^{-1} = \left(\begin{array}{c|c} \mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{train}, \theta_1) & \mathbf{K}_1(\mathbf{X}_1^{query}, \mathbf{X}_1^{train}, \theta_1) \\ \hline \mathbf{K}_1(\mathbf{X}_1^{train}, \mathbf{X}_1^{query}, \theta_1) & \mathbf{K}_1(\mathbf{X}_1^{query}, \mathbf{X}_1^{query}, \theta_1) \end{array} \right)^{-1} + \left(\begin{array}{c|c} 0_{ntrain,ntrain} & 0_{ntrain,nquery} \\ \hline 0_{nquery,ntrain} & \left(\mathbf{K}_2(\mathbf{X}_2^{query}, \mathbf{X}_2^{query}, \theta_1) + \mathcal{V}_2^{pred}\right)^{-1} \end{array} \right). \tag{4.10}$$

The "Hybrid" method should respect the conditional independence associated with co-training, as it consists of a sum over the precision matrices of each view. It should be able to act as a generative model to predict at arbitrary test data, albeit with the transductive property that the kernel necessitates. Unlike the previous two methods, this model incorporates maximal covariance information from both agents, so we might *a priori* expect it to perform more effectively. However, we can expect building and inverting the transductive kernel to be more computationally expensive than either of the previous methods.

4.2.3 Uncertainty Integration

It is not clear how to integrate the uncertainty associated with the predictive distributions. Here we pursue three different methods. Firstly, we could not use the uncertainty at all, and treat the mean of the predictions as the data value. Specifically, the matrix $\mathcal{V}_2^{pred}$ as mentioned above would simply be a matrix of zeros.

Secondly, we could use the marginal variance at each prediction, and add it to the corresponding diagonal of the covariance matrix. Formally, in the above notation, $\mathcal{V}_2^{pred} = \text{diag}[\text{Var}(\mathbf{y}_2^{pred})]$.

Thirdly, we could use the full predictive covariance and add it to the corresponding submatrix in the covariance. In this case, $\mathcal{V}_2^{pred} = \text{Cov}(\mathbf{y}_2^{pred})$. It would be expected that including more information about the predictive distribution to improve performance, but also reveal more about the private data covariates.

4.3 Experiments with Simulated Data

4.3.1 Experimental Features

There are several features of the experimental design that we would like to vary, beyond the above discussed combination methods. A one-dimensional exponentiated quadratic covariance function was used for the GP kernel in every experiment.

4.3.1.1 Data Generation

We proceed with an experimental design, in which there are two views, each of which consists of a single covariate. The covariates and response variable are generated from a known covariance matrix, such that the true model is a linear model with a known correlation structure. Five data properties are varied, which are explained here.

Firstly, the correlation strength $r_{\mathbf{Xy}}$ between the covariates and the response variable can be changed. It is possible that the methods will behave differently for different correlation strengths. Here the two views are generated with equal correlation strength with the response variable, but with varying correlation values equal to 0.5, 0.75, 0.9 and 0.99.

Secondly, the correlation between the different covariates can be changed. Either we can enforce conditional independence by imposing zeros in the generative precision matrix, or we can assign the correlation a value of 1, such that the two views are effectively the same variable. It would be expected the methods which are more explicitly based on the assumptions of co-training (i.e. "Hybrid" and "Blocky") to perform better with the conditional independence assumption enforced.

Thirdly, the correlated covariates can be masked for one to be positive and the other to be negative. Given their positive correlation, this will make each of the initial models to be experts in their regions of covariate space. Consequently, any increase in power associated with combining the expertise of the models should be stronger in this situation.

Fourthly, we can change the number of private data points, using values of n_{obs} equal to 25, 50 and 200: It would intuitively make sense for models that have seen fewer data to gain more from learning from one another. We simulated 500 test data points and 500 unlabelled query points for each experiment. Co-training consisted of 25 rounds of each GP alternately querying single data points using uncertainty sampling: each model selected the unlabelled point upon which it evaluated the highest predictive variance.

Finally, the random seed generating the data can be changed, providing ten randomised iterations of the experiment.

4.3.1.2 Explainable Covariance

The data are generated from a known covariance matrix. As such, it is possible to establish the maximum possible explainable variance of the response variable, given the information present in the covariates, either separately or jointly. Given that in these experiments the covariates have identical correlation properties with the response variable then the individual explainable variance is identical for each.

Here are presented the unexplainable standard deviations of the response variable in each experimental situation, as an example of what models built on the data can achieve in optimal circumstances. The relevant experimental variables are:

- $r_{\mathbf{Xy}}$: the correlation strength between the covariates and the response variable.
- Conditional independence: whether the precision between the covariates has been set to zero.
- Masking status: whether each model sees data from all of covariate space.

Table 4.1 demonstrates the the square root of the variance remaining in the response variable after each covariate has optimally individually removed the variance that it is capable of explaining.

Masking	Cond. ind.	Correlation Strength 0.5	0.75	0.9	0.99
No	No	0.866	0.661	0.436	0.141
	Yes	0.866	0.661	0.436	0.141
Yes	No	0.935	0.848	0.771	0.714
	Yes	0.935	0.848	0.771	0.714

Table 4.1: The unexplainable standard deviation in the response variable $\mathbf{y}$, given the covariates' independent predictions.

Table 4.2 demonstrates the the square root of the variance remaining in the response variable after the covariates have optimally jointly removed the variance that they are capable of explaining.

Masking	Cond. ind.	Correlation Strength 0.5	0.75	0.9	0.99
No	No	0.866	0.661	0.436	0.141
	Yes	0.775	0.529	0.324	0.100
Yes	No	0.866	0.661	0.436	0.141
	Yes	0.775	0.529	0.324	0.100

Table 4.2: The unexplainable standard deviation in the response variable $\mathbf{y}$, given the covariates' joint predictions.

4.3.2 Results

4.3.2.1 Pre Co-training Results

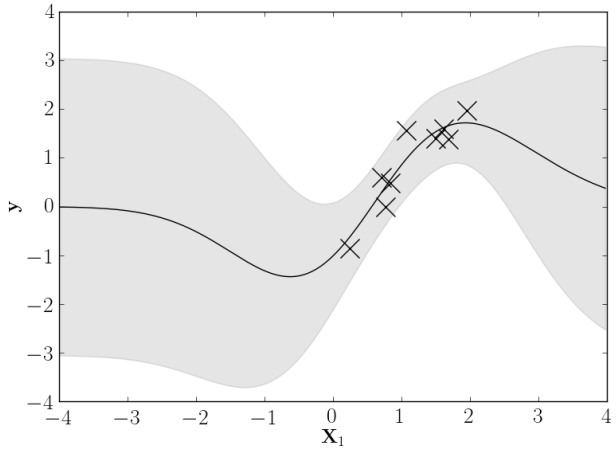
It is useful to see the predictive performance of the models trained on each view, after the initial hyperparameter training but before the co-training and exchange of information. In Table C.1 in Appendix C we present the means and standard deviations between runs of the root mean square error within each run. The models perform broadly as expected, with the GPs with strong correlation, more data and less masking producing more accurate predictions.

4.3.2.2 Comparison of Combination Methods

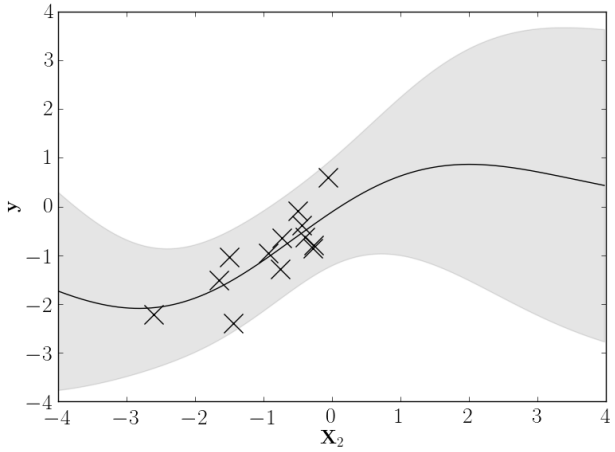
We see in Figure 4.3 two diagrams representing a typical exchange of queries between two statisticians. We see that the data queries are drawn from regions where the GP trained on private data exhibits high posterior variance.

Table 4.3 presents the performance of the models after hyperparameter learning and co-training. The results are the root mean square errors within each run, presented as a mean and standard deviations between runs, taken over all of the experimental variables except for the view combination methods:

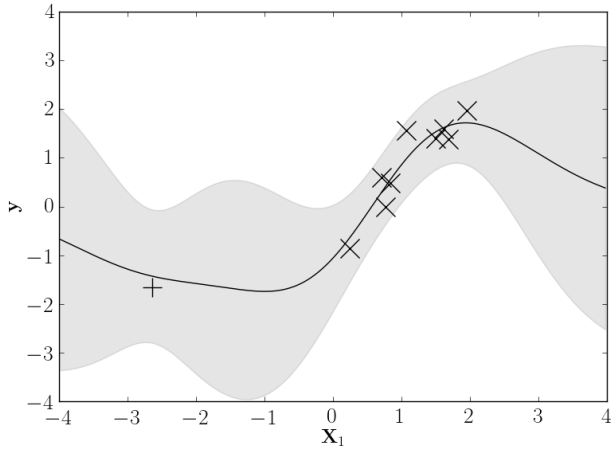
The most immediately striking result is that the "Blocky" combination method performs very poorly. Considering that the response variable is normalised to have unit variance, then it is not possible to defend the properties of the predictions generated by this method. This is



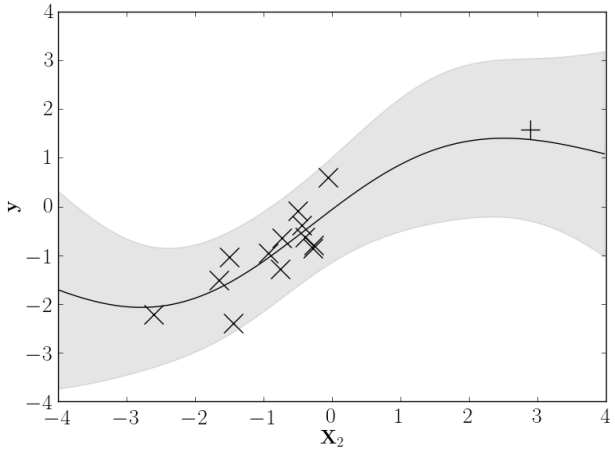
(a) View 1 before co-training



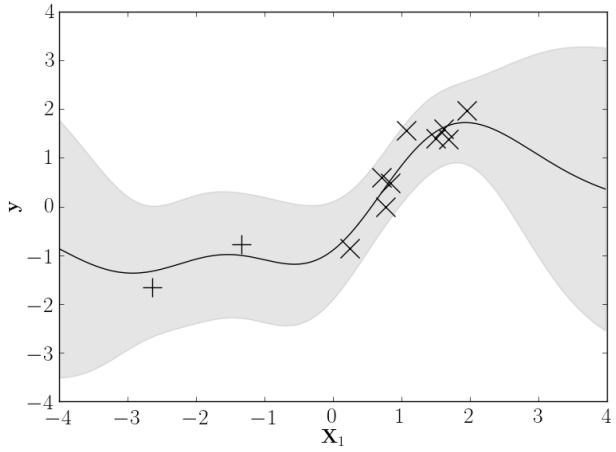
(b) View 2 before co-training



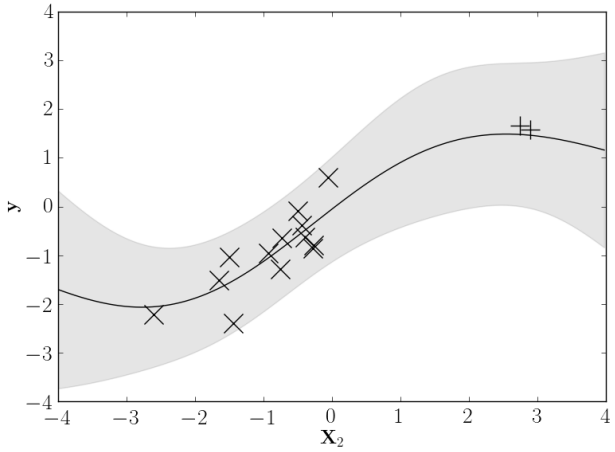
(c) View 1 after one co-training query



(d) View 2 after one co-training query



(e) View 1 after two co-training queries



(f) View 2 after two co-training queries

Figure 4.3: A visual demonstration of the exchange of queries between two statisticians. The private labelled data is indicated by crosses and the queried data as pluses. We also see the posterior mean and 95% credible interval of the GP trained on updated data. The experiment was performed with masked data, with 25 private labelled data, $r_{\mathbf{Xy}} = 0.9$ and conditional independence enforced.

not a complete surprise: the method is not consistent in its use of covariance functions, so the predictions are not guaranteed to be drawn from a consistent Gaussian distribution.

Combination Method		
"Blocky"	"Noisy"	"Hybrid"
1.67 ± 3.03	$\mathbf{0.574} \pm 0.286$	0.593 ± 0.280

Table 4.3: The test RMSEs of the GPs with trained hyperparameters after co-training, presented as means and standard deviations between runs. Results are stratified by combination method.

In general, it was observed that the act of co-training improved the performance of the models, with the exception of the "Blocky" combination method, which substantially worsened the predictive RMSE of the models.

We observe that the "Noisy" and "Hybrid" methods have similar RMSE on the test data, with the "Noisy" method exhibiting marginally more accuracy on average. This is perhaps surprising, as the "Hybrid" method uses more covariance information, and is more closely related to the previously derived work on the topic. The standard deviations of the errors are very similar for each combination method.

When the difference in variance between "Noisy" and "Hybrid" methods is examined, it becomes clear that certain subsets of the experimental results contribute more to the difference in average predictive power. We see in Table 4.4 that there is an apparent interaction between masking, combination method and correlation strength. The difference in performance between "Noisy" and "Hybrid" methods is most marked when using masked data with high correlation strength.

		Combination Method		
Masking	$r_{\mathbf{Xy}}$	"Blocky"	"Noisy"	"Hybrid"
No	0.5	1.04 ± 0.304	0.901 ± 0.0502	$\mathbf{0.895} \pm 0.0669$
	0.75	1.32 ± 1.10	$\mathbf{0.693} \pm 0.0485$	0.703 ± 0.0633
	0.9	1.71 ± 2.91	$\mathbf{0.458} \pm 0.034$	0.472 ± 0.0536
	0.99	1.86 ± 2.93	$\mathbf{0.147} \pm 0.0134$	0.151 ± 0.0186
Yes	0.5	1.07 ± 0.425	0.935 ± 0.0549	$\mathbf{0.924} \pm 0.0626$
	0.75	1.36 ± 1.26	$\mathbf{0.752} \pm 0.0917$	0.765 ± 0.0892
	0.9	1.74 ± 2.09	$\mathbf{0.510} \pm 0.0808$	0.557 ± 0.129
	0.99	3.25 ± 6.74	$\mathbf{0.200} \pm 0.0834$	0.279 ± 0.183

Table 4.4: The test RMSEs of the GPs with trained hyperparameters after co-training, presented as means and standard deviations between runs. Results are stratified by combination method, $r_{\mathbf{Xy}}$ and masking status.

We see that with unmasked data then "Noisy" combination is marginally more successful than "Hybrid" data combination. With masked data, we see that "Noisy" combination is substantially more successful than "Hybrid" combination at high correlation strength. We can also see that the standard deviation of the "Hybrid" standard errors in this regime is also much higher than the "Noisy" method. This suggests that the mean may be skewed by outliers.

Experience in training "Hybrid" models in this regime suggests that hyperparameter inference is quite challenging.

4.3.2.3 Relation with Data Correlation Structure

The experiments were performed with views where the covariates had varying correlation structure. Half of the experiments were performed with view covariates that were conditionally independent given the response variable, and the other half used two views that were very highly correlated, effectively being the same variable and containing the same information.

Each of the combination methods makes different assumptions regarding the data structure: the "Hybrid" structure relies on the co-training assumptions of conditional independence, whereas the "Noisy" method uses a single covariance over all of the data, perhaps more suiting the data with very similar covariates.

	Combination Method		
Cond. ind.	"Blocky"	"Noisy"	"Hybrid"
No	1.65 ± 2.29	$\mathbf{0.562} \pm 0.28$	0.579 ± 0.281
Yes	1.69 ± 3.62	$\mathbf{0.587} \pm 0.291$	0.608 ± 0.278

Table 4.5: The test RMSEs of the GPs with trained hyperparameters after co-training, presented as means and standard deviations between runs. Results are stratified by combination method and conditional independence status.

The results presented in Table 4.5 describing the performance of the two combination methods with the data structure do not show a change in the difference between methods when comparing the two correlation structures.

		Combination Method		
Masking	Cond. ind.	"Blocky"	"Noisy"	"Hybrid"
No	No	1.41 ± 1.19	$\mathbf{0.543} \pm 0.278$	0.552 ± 0.284
	Yes	1.55 ± 2.82	$\mathbf{0.557} \pm 0.288$	0.558 ± 0.281
Yes	No	1.89 ± 2.99	$\mathbf{0.582} \pm 0.28$	0.606 ± 0.275
	Yes	1.82 ± 4.27	$\mathbf{0.617} \pm 0.291$	0.657 ± 0.266

Table 4.6: The test RMSEs of the GPs with trained hyperparameters after co-training. Results are stratified by combination method, conditional independence status and masking status.

Some further structure becomes clear in Table 4.6 when further separating on the masking status. It can be seen that when the data are not masked, then the difference between the methods disappears when conditional independence is applied: this in accordance with the assumptions made in deriving the methods, as we would expect the "Hybrid" combination method to be better suited to conditionally independent data. A similar effect is not observed when the data are masked, but that could be a result of the difficult hyperparameter optimisation in the masked regime, rather than a fundamental property of the methods. This is implied by the high variance of the "Noisy" method with masked and conditionally independent data, suggesting the presence of a few poorly performing outliers.

4.3.2.4 Comparison of Uncertainty Integration

Another main question of interest concerns how best to incorporate the uncertainty of the other GP's predictions into the model. The experiments explored three approaches: ignoring the uncertainty completely, including the marginal variances or including the full covariance matrix of the predictions. It would be intuitively sensible that including more information about the distributions would generate a more powerful model.

However, there is an increased privacy risk associated with sharing more information, so if the extra uncertainty information does not prove useful, then it may be most appropriate not to share it. We might expect the effects of adding uncertainty to appear more strongly when the data have been masked, given that the two models will have strongly differing confidences in their prediction.

		Combination Method		
Masking	Covariance	"Blocky"	"Noisy"	"Hybrid"
No	None	1.13 ± 0.816	$\mathbf{0.551} \pm 0.284$	0.561 ± 0.286
	Marginal	1.07 ± 0.72	$\mathbf{0.548} \pm 0.282$	0.554 ± 0.281
	Full	2.25 ± 3.47	$\mathbf{0.549} \pm 0.283$	0.550 ± 0.281
Yes	None	1.44 ± 1.75	$\mathbf{0.616} \pm 0.27$	0.646 ± 0.273
	Marginal	1.26 ± 1.31	$\mathbf{0.595} \pm 0.29$	0.630 ± 0.27
	Full	2.86 ± 5.87	$\mathbf{0.588} \pm 0.298$	0.619 ± 0.271

Table 4.7: The test RMSEs of the GPs with trained hyperparameters after co-training. Results are stratified by combination method, uncertainty integration method and masking status.

We again observe in Table 4.7 that the "Blocky" method performs extremely poorly as previously.

We observe that the predictive power increases with the integration of more uncertainty information, with one marginal exception. We can also see that the uncertainty information is more valuable when the data are masked: this is intuitively sensible, as each of the GPs will have greater differences in predictive uncertainty when they are experts on different regions of covariate space. It appears that a substantial amount of the increase in predictive performance is brought about by sharing the marginal variances.

4.4 Differential Privacy

The above work concerns itself with the question of how to perform co-training between Gaussian Processes without explicitly sharing the data. This is not sufficient to make claims of differential privacy, wherein the individual data cannot be identified by individual queries. For example, if one of the agents systematically queries the other for the value of the mean over all of the covariate space, then they can identify low-variance locations and deduce the value of the private response data at these locations. The mechanism for guaranteeing the differential privacy of the response variables is covered in Section 1.4. We follow the example of (Smith

et al., 2016), in which they choose $\epsilon = 1$ and $\delta = 0.00625$, giving a scaling factor of 0.42. Similarly, we use a value of $\Delta = 1$, as the response variable has unit variance.

4.4.1 Differential Privacy Results

The same set of experiments was run as previously, except with the scaled samples from the GP prior added to the communicated predictive mean, maintaining the differential privacy of the response variable in the communication between GPs.

4.4.1.1 Pre Co-training Results

Again, we present in Table C.2 in Appendix C the predictive performance of the models trained on each view, after the initial hyperparameter training but before the co-training and exchange of information. The models again exhibit very similar behaviour to the previous pre-co-training results. This is to be expected, as the new differentially private co-training methods have not been used at this point in the algorithm.

4.4.1.2 Comparison of Combination Methods

Figure 4.4 represents differentially private exchange of queries in co-training. We again see that the data queries are drawn from regions where the GP trained on private data exhibits high posterior variance, although they exhibit less fidelity to the linear trend than the non-differentially private queries presented in Figure 4.4.

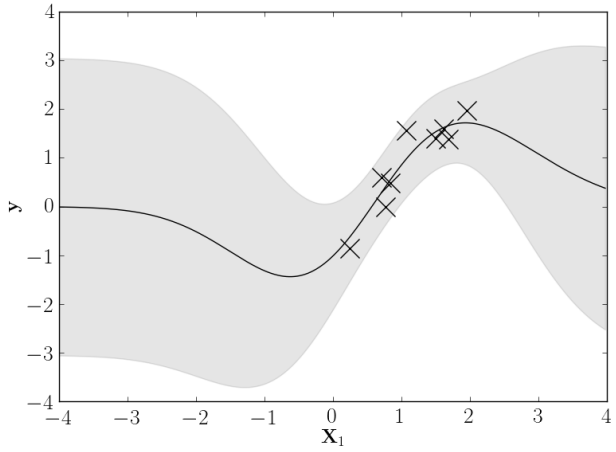
Combination Method		
"Blocky"	"Noisy"	"Hybrid"
4.11 ± 20.6	$\mathbf{0.672} \pm 0.256$	0.758 ± 0.274

Table 4.8: The test RMSEs of the GPs with trained hyperparameters after differentially private co-training, presented as means and standard deviations between runs. Results are stratified by combination method.

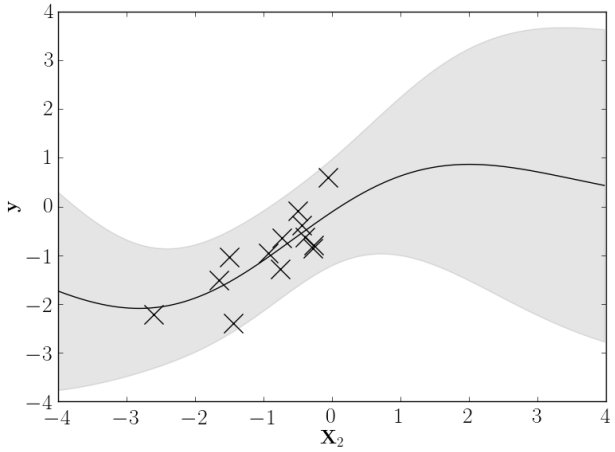
Table 4.8 compares the behaviour of the combination methods under differential privacy, averaging over every other experimental variable.

As before, we see that the "Blocky" method performs very poorly. However, we observe that the "Noisy" combination performs substantially better than the "Hybrid" method on average for the differentially private experiments, whereas previously there only existed a marginal difference in the mean test RMSE. It remains surprising that the "Noisy" method is much more successful, as it uses less covariance information than the "Hybrid" combination method.

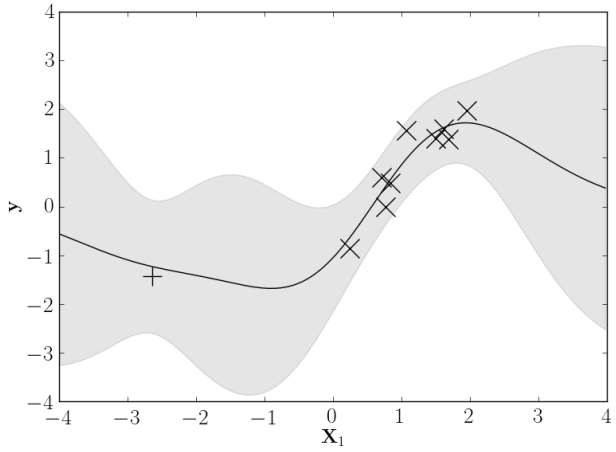
We observe that, in contrast to the previous experiments, differentially private co-training did not always improve performance for the "Noisy" and "Hybrid" methods. It would be expected for the differentially private experiments to not perform as well as the others, but it would still be desirable for the contribution of co-training to always be positive.



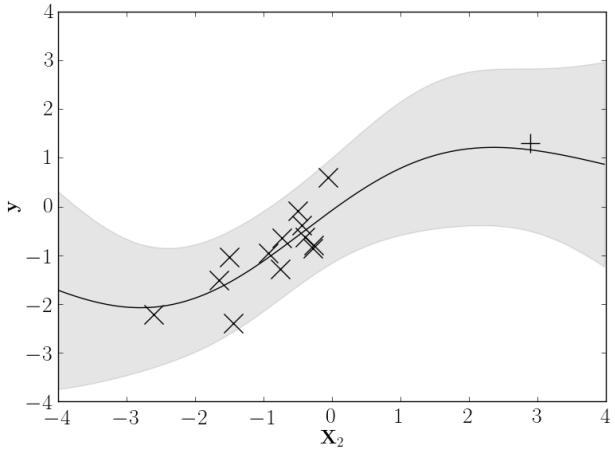
(a) View 1 before co-training



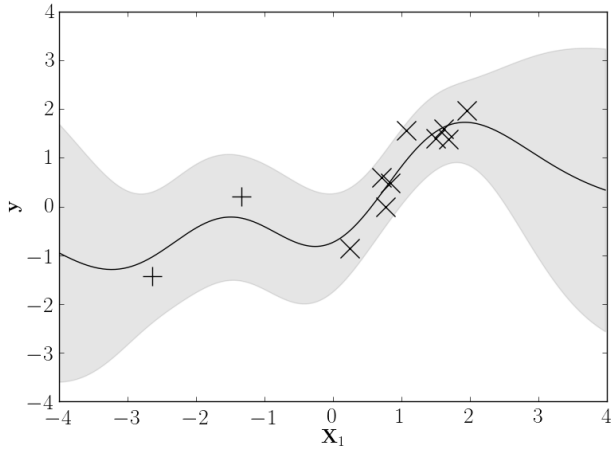
(b) View 2 before co-training



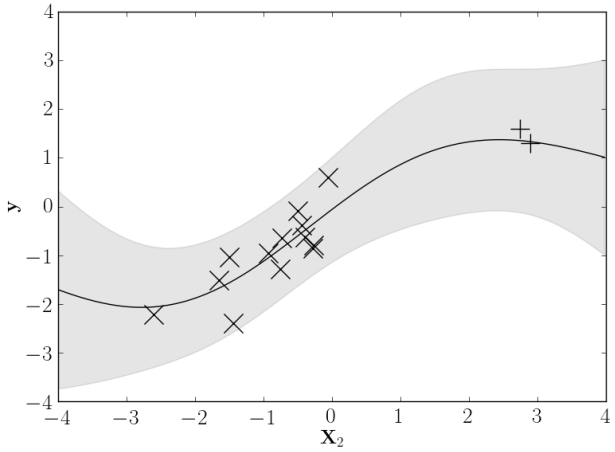
(c) View 1 after one co-training query



(d) View 2 after one co-training query



(e) View 1 after two co-training queries



(f) View 2 after two co-training queries

Figure 4.4: A visual demonstration of the differentially private exchange of queries between two statisticians. The private labelled data is indicated by crosses and the queried data as pluses. We also see the posterior mean and 95% credible interval of the GP trained updated on the observed data. The experiment was performed with masked data, with 25 private labelled data, $r_{\mathbf{Xy}} = 0.9$ and conditional independence enforced.

4.4.1.3 Comparison of Combination Method, Masking and Correlation Strength

Masking	$r_{\mathbf{Xy}}$	Combination Method "Blocky"	"Noisy"	"Hybrid"
No	0.5	1.28 ± 1.24	0.905 ± 0.052	$\mathbf{0.900} \pm 0.0694$
	0.75	2.68 ± 5.14	$\mathbf{0.711} \pm 0.0638$	0.738 ± 0.0996
	0.9	4.78 ± 12.2	$\mathbf{0.513} \pm 0.0924$	0.621 ± 0.241
	0.99	15.3 ± 54.8	$\mathbf{0.464} \pm 0.324$	0.682 ± 0.51
Yes	0.5	1.14 ± 0.513	0.948 ± 0.0688	$\mathbf{0.937} \pm 0.077$
	0.75	1.49 ± 1.36	$\mathbf{0.787} \pm 0.11$	0.838 ± 0.158
	0.9	2.01 ± 2.23	$\mathbf{0.606} \pm 0.156$	0.697 ± 0.197
	0.99	4.21 ± 7.93	$\mathbf{0.446} \pm 0.304$	0.650 ± 0.317

Table 4.9: The test RMSEs of the GPs with trained hyperparameters after differentially private co-training, presented as means and standard deviations between runs. Results are stratified by combination method, $r_{\mathbf{Xy}}$ and masking status.

Previously, we observed that a substantial proportion of the average difference between the performance of the combination methods was drawn from a particular subset of experimental conditions. In Table 4.9 we break the results down by the same structure to see if similar structure existed under differential privacy. We observe that the methods perform similarly with very weak correlation, with the difference in performance becoming very marked as correlation increases, both with and without masked data.

We also observe that the variance of the RMSE greatly increases as the correlation increases, suggesting that the average performance is being thrown off by some very inaccurate runs. This could be a result of more challenging inference: if the marginal likelihood is very highly peaked by more data, it may be more difficult to optimise the hyperparameters and kernel variances.

4.4.1.4 Comparison of Combination Method and Correlation Structure

Cond. ind.	Combination Method "Blocky"	"Noisy"	"Hybrid"
No	3.22 ± 9.42	$\mathbf{0.663} \pm 0.251$	0.754 ± 0.284
Yes	5.01 ± 27.6	$\mathbf{0.682} \pm 0.260$	0.761 ± 0.264

Table 4.10: The test RMSEs of the GPs with trained hyperparameters after differentially private co-training, presented as means and standard deviations between runs. Results are stratified by combination method and conditional independence status.

Given the assumptions of conditional independence used in deriving the "Hybrid" method, we might expect the performance difference between "Noisy" and "Hybrid" to be less marked when using conditionally independence data. We observe in Table 4.10 that the difference between the two is slightly smaller when conditional independence is enforced. However, the "Noisy" method still performs significantly better under either data correlation structure.

4.4.1.5 Comparison of Combination Method, Correlation Structure and Masking

Masking	Cond. ind.	Combination Method "Blocky"	"Noisy"	"Hybrid"
No	No	4.28 ± 12.3	$\mathbf{0.639} \pm 0.236$	0.744 ± 0.329
	Yes	7.75 ± 38.6	$\mathbf{0.658} \pm 0.255$	0.727 ± 0.282
Yes	No	2.16 ± 4.87	$\mathbf{0.686} \pm 0.264$	0.765 ± 0.23
	Yes	2.26 ± 3.77	$\mathbf{0.707} \pm 0.262$	0.796 ± 0.24

Table 4.11: The test RMSEs of the GPs with trained hyperparameters after differentially private co-training, presented as means and standard deviations between runs. Results are stratified by combination method, conditional independence status and masking status.

If the data are further stratified on masking status as in Table 4.11, we notice that the "Hybrid" method performs markedly better when the conditional independence criteria are enforced and the data are not masked, but performs worse under conditional independence and the data are masked. This suggests that the situation is more complex than the "Hybrid" method performing better under conditional independence, although it is not clear what other effects may be influencing the results.

4.4.1.6 Comparison of Combination Method, Masking and Uncertainty Integration

Masking	Covariance	Combination Method "Blocky"	"Noisy"	"Hybrid"
No	None	2.13 ± 3.48	$\mathbf{0.662} \pm 0.248$	0.746 ± 0.323
	Marginal	1.97 ± 3.23	$\mathbf{0.647} \pm 0.244$	0.739 ± 0.304
	Full	14.0 ± 48.5	$\mathbf{0.636} \pm 0.245$	0.721 ± 0.291
Yes	None	1.76 ± 1.81	$\mathbf{0.731} \pm 0.254$	0.794 ± 0.241
	Marginal	1.41 ± 1.29	$\mathbf{0.690} \pm 0.261$	0.780 ± 0.232
	Full	3.46 ± 7.03	$\mathbf{0.668} \pm 0.27$	0.768 ± 0.232

Table 4.12: The test RMSEs of the GPs with trained hyperparameters after differentially private co-training, presented as means and standard deviations between runs. Results are stratified by combination method, uncertainty integration method and masking status.

We see in Table 4.12 that including any predictive uncertainty improves the performance of the models, more clearly when masked, as might be expected. Consequently, if we are solely interested in preserving the integrity of the privacy of the response variable, then we can add all the relevant uncertainty information without concern.

4.5 Experiments with Real Data

We follow the simulated data with a real data example. We used the Combined Cycle Power Plant ("CCPP") data set from the UC Irvine database. The response variable is the net hourly energy output of a power plant, to be predicted using four covariates: the ambient

temperature, ambient pressure, relative humidity and the exhaust vacuum. The covariates and response variable were scaled to have mean zero and variance one.

The views were generated by dividing the covariates into two sets of two. Given that this is real data, we cannot enforce conditional independence on the data precision matrix, so we should not assume that the conditional assumptions relevant to co-training will be valid in this context.

Masking was achieved by running a k-means algorithm on the covariates to generate two clusters. This successfully separated the covariate space into two regions, such that a model trained on each would be an expert on that proportion of the data.

All of the simulations used an ARD exponentiated quadratic kernel. Similarly, they all used 250 test data points, 250 unlabelled data points for querying, and co-training consisted of 25 rounds of alternately querying individual data points. The experiments explored the following designs:

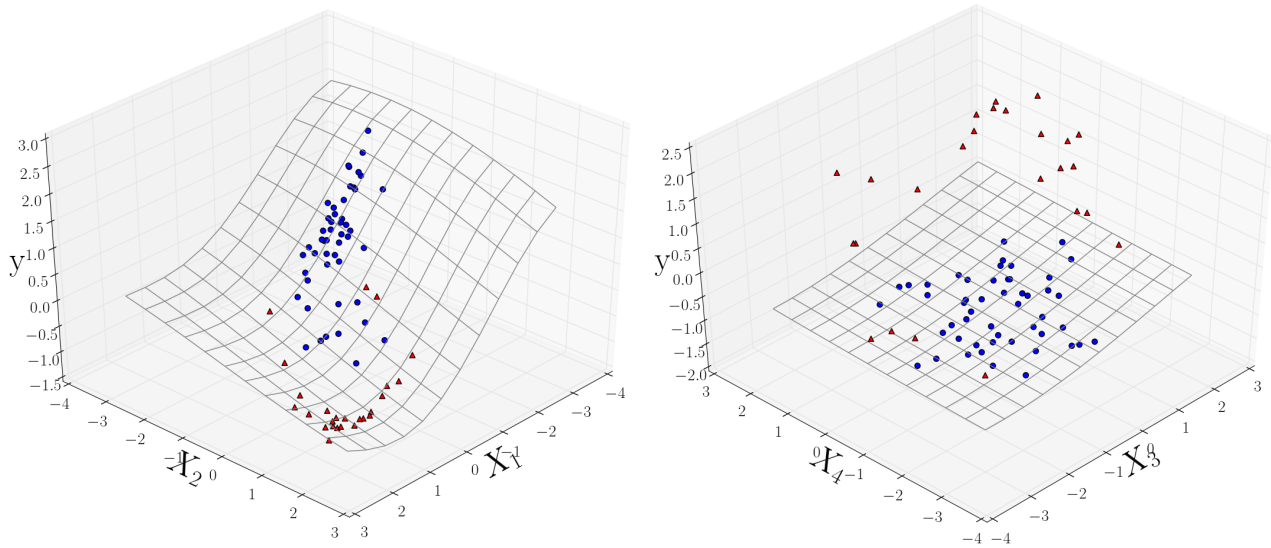
- The three combination methods of "Blocky", "Noisy" and "Hybrid".
- The number of labelled data observed in the training data n_{obs}, with values of 25, 50, 100 and 200.
- The masking status of the data: whether each of the views has been trained on specific clusters of the covariates, or the whole data space.
- Whether the predictive uncertainty is incorporated into the co-training algorithm. The options involved were to disregard it, add the marginal variance or the full predictive covariance.

4.5.1 Pre Co-Training Performance

The test RMSEs of each of the views on the test data, with optimised hyperparameters but no co-training, are presented in Table 4.13.

		View	
Masking	n_{obs}	1	2
No	25	0.33	0.872
	50	0.311	0.838
	100	0.305	0.804
	200	0.305	0.806
Yes	25	0.505	1.17
	50	0.343	1.18
	100	0.341	1.12
	200	0.313	1.14

Table 4.13: The test RMSEs of the GPs with hyperparameters trained on CCPP data set, but before co-training. Results are stratified by view, n_{obs} and masking status.



(a) View 1

(b) View 2

Figure 4.5: A visual demonstration of the exchange of queries between two statisticians. The private labelled data is indicated by blue circles and the queried data as red triangles. We also see the posterior mean of the GP trained on the private labelled data represented as a wire grid. The experiment was performed with masked data and 50 private labelled data.

We see that view 1 is much more successful in prediction than view 2. We notice that view 2 is particularly poor when learning from masked data, returning RMSE values of greater than one. This seems counter-intuitive when the response variable is normalised to have variance one, but if the response variables of the two clusters of data have significantly different means, then such high values of the test RMSE are possible.

4.5.2 Post Co-Training Results without Differential Privacy

An exchange of queries between statisticians without differential privacy is presented in Figure 4.5. We see that the data queries are drawn from regions where there is less training data, and consequently the original GP has higher variance.

Having examined the results, it appeared that including the full predictive covariance in co-training generated the lowest RMSE on average, so those results are presented in Table 4.14.

We observe that the "Blocky" method results in very poor performance, as in the experiments with simulated data.

It is further clear that co-training is helpful for the weaker view 2, especially when it has not seen much labelled data and is trained using masked data. However, co-training rarely improves the performance of the stronger view 1, and can make the test RMSE substantially worse.

		Combination Method					
		"Blocky"		"Noisy"		"Hybrid"	
		View		View		View	
Masking	n_{obs}	1	2	1	2	1	2
No	25	1.00	1.57	**0.403**	0.806	0.473	**0.803**
	50	0.970	1.62	**0.379**	**0.806**	0.515	0.818
	100	2.00	1.14	**0.307**	0.810	0.362	**0.809**
	200	1.42	1.47	**0.306**	0.810	0.324	**0.808**
Yes	25	0.611	4.76	**0.498**	1.08	0.643	**0.898**
	50	2.16	2.40	**0.345**	0.998	0.472	**0.912**
	100	1.30	2.23	**0.334**	0.96	0.540	**0.934**
	200	4.88	**0.883**	**0.327**	1.07	0.416	1.04

Table 4.14: The test RMSEs of the GPs with hyperparameters trained on CCPP data set with non-differentially private co-training. Results are stratified by view, n_{obs}, Masking status and combination method.

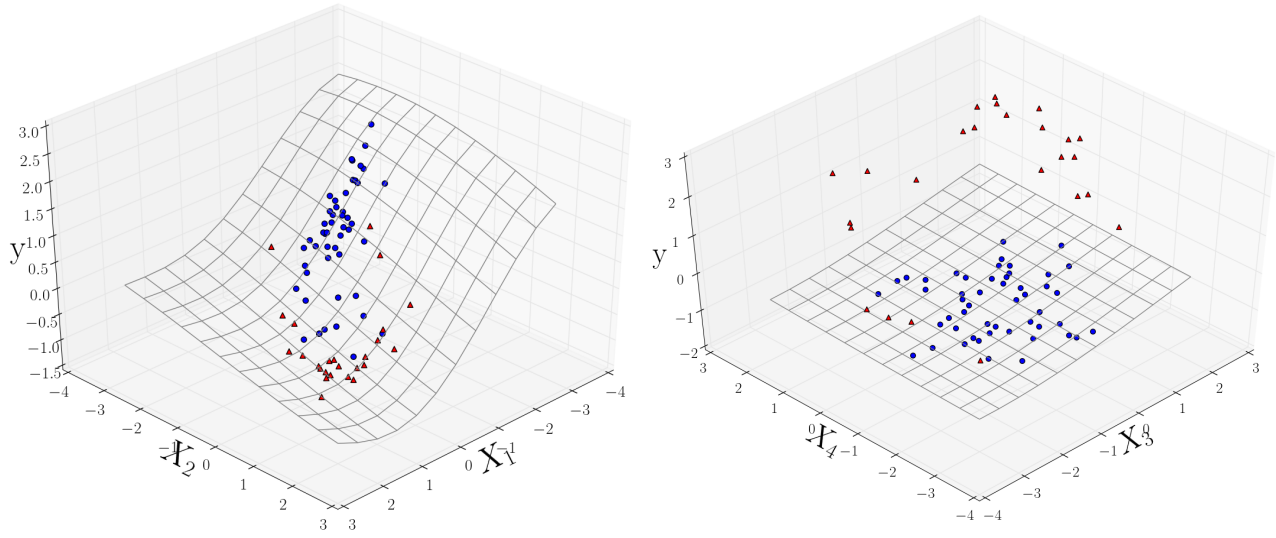
Comparing combination methods, the "Noisy" combination method is more successful than "Hybrid", with the exception of view 2 working with masked training data. It is possible that simply capturing the mean of each cluster would be sufficient to substantially improve the performance in this circumstance, which the "Hybrid" method may be communicating more effectively.

4.5.3 Post Co-Training Results with Differential Privacy

Figure 4.6 represents an exchange of queries between statisticians while enforcing differential privacy of the response variable. We again see that the data queries are drawn from regions where there is less training data, with greater variation in the values of the queried response variable in comparison to Figure 4.5.

		Combination Method					
		"Blocky"		"Noisy"		"Hybrid"	
		View		View		View	
Masking	n_{obs}	1	2	1	2	1	2
No	25	1.78	1.48	**0.356**	**0.804**	0.551	0.903
	50	0.817	2.96	**0.352**	0.807	0.590	**0.836**
	100	4.23	2.90	**0.308**	0.819	0.336	**0.810**
	200	1.58	1.31	**0.306**	0.830	0.316	**0.821**
Yes	25	1.61	6.33	**0.516**	1.11	0.630	**0.869**
	50	4.12	6.31	**0.460**	0.999	0.558	**0.922**
	100	2.51	6.10	**0.436**	**0.963**	0.633	1.14
	200	6.92	**0.914**	**0.327**	1.09	0.382	1.04

Table 4.15: The test RMSEs of the GPs with hyperparameters trained on CCPP data set with differentially private co-training. Results are stratified by view, n_{obs}, masking status and combination method.



(a) View 1

(b) View 2

Figure 4.6: A visual demonstration of the differentially private exchange of queries between two statisticians. The private labelled data is indicated by blue circles and the queried data as red triangles. We also see the posterior mean of the GP trained on the private labelled data represented as a wire grid. The experiment was performed with masked data and 50 private labelled data.

We observe in Table 4.15 that the results with differentially private co-sharing provide an improvement on the pre co-training results in certain circumstances. Specifically, the "Noisy" method improves the performance of view 2 both with and without masking, and the "Hybrid" method helps when view 2 is masked.

These results are consistent with the results from the previous section: Co-training is often more beneficial for the less informative view, and is more valuable when the data have been masked between views. We again observe that the "Noisy" method is generally more successful on real data.

4.6 Conclusions and Future Work

4.6.1 Combination Methods

When not enforcing differential privacy, it is observed that the "Noisy" combination will generally outperform the "Hybrid" method in the experimental contexts explored, although the difference can be reduced or reversed if there is very weak correlation, or the data exhibit conditional independence given the response variable. It is possible that the mean of the predictive errors is skewed by a minority of poorly performing, badly optimised models: this is suggested by the higher standard deviation of the RMSE in certain experimental regimes.

When differential privacy of the response variable is enforced, the "Noisy" method consistently outperforms the "Hybrid" method in almost every experimental regime, making it the preferable option in these circumstances.

In any situation, the "Blocky" method performs very poorly, and is not at all competitive compared to the other two combination methods.

4.6.2 Predictive Uncertainty Integration

Whether enforcing differential privacy or not, it is observed that including uncertainty information of the predictive distributions improves the models' performance. It is seen that a substantial increase is brought about by including only the marginal variances: it is dependent on the context as to whether the privacy risk associated with sharing more information is suitable. There is no information about the response variable in the predictive uncertainty, so it will not affect the privacy of the response variable to share predictive covariance information.

If privacy concerns are compelling us to add extra variation to the shared predictive mean, then it is possible that it is necessary to add to the shared predictive variance in order to characterise the uncertainty properly. The treatment of the predictive uncertainty is an important aspect of co-training, and it is possible that by underestimating the predictive variance we are performing sub-optimal model combination. Such work has not been explored in previous work on differentially private Gaussian Processes, and would be an interesting new research direction.

It is also possible that performing fully Bayesian inference over the kernel hyperparameters would help to better characterise the predictive uncertainty of the models. This would represent a more expensive inference choice, but should add an appropriate amount of variance that would facilitate co-training between models.

4.6.3 Differential Privacy of Covariates

This work has explored formally enforcing differential privacy of the response variable: it would also be of interest to formally explore the privacy of covariates. This was respected here by assuming that each expert did not have access to the others' covariates, and would not share the covariance of their latent variable over their private, labelled data set. There are other stronger ways of treating the privacy of the covariates, either by using methods to strictly ensure differential privacy as pursued in (Smith et al., 2016), or reconsidering what covariance information would be appropriate to share during combination.

Chapter 5

Conclusion

The task of increasing the number of inducing points used by a variational Gaussian Process is in principle a productive one: drawing more information from across the covariate space will provide a more informative variational approximation. However, the corresponding increase in variational parameters in a stochastic variational GP will make the model optimisation more challenging.

We can see in Tables A.9 and A.10 that even a standard SVIGP will struggle to optimise properly on the 2-dimensional Jutland data set for large values of m. As such, we should perhaps proceed with caution when introducing modifications to the standard variational framework: it is difficult to know *a priori* which configuration of variational distribution and optimisation parameters will be successful for a given data set.

Considering the Tucker Variational GP, the structured variational distribution has significant potential: it can capture information from huge numbers of implicit inducing points that cover the covariate space, while maintaining a specifiable and manageable number of variational parameters.

However, it seems that the product structure of the Tucker decomposition may introduce some instability to the inference procedure, resulting in some very extreme outliers for the test RMSE. It is possible that a blocky, additive style model over subsets of dimensions will reduce some of the instability associated with large products, while also avoiding the problem of dimensionally-cursed variational representations.

For the TVGP, it is likely that trying further optimisation schedules would help stabilise the inference: AdaDelta and Adam can be more successful than Adagrad, and require less tuning. Similarly, results suggest that being more generous with the batch size will contribute positively to the optimisation.

The new $O(pm_0)$ covariance representation used in Chapter 2 has not been thoroughly interrogated: it does not seem to have compromised the hyperparameter inference, but it would be interesting future work to examine the resulting predictive variances to ensure that they are well-calibrated. The "Blitzkriging" approximation has demonstrated that it is possible to successfully impose Kronecker structure on the variational covariance (Nickson et al., 2015), so it would only be the diagonal variational representation that may cause a problem.

Appraising the Randomised Numerics Variational GP, it seems that heavily parametrised variational methods need all the help they can get: it is possible that increasing the number of computationally feasible inducing points by introducing gradient noise is tantamount to posing a more challenging problem with less powerful tools.

It is likely that the optimisation schedule would appreciate further tuning. Using AdaDelta or Adam could improve the situation, as might increasing the batch size and similar parameters. It is possible that the optimisation will need to be customised for each new problem, which may compromise the RNVGP's general applicability.

For this work, the $O(m)$ covariance representation is necessary for the $O(m^2)$ computation to work. Further work would involve ensuring that the predictive variance accurately represents the model's uncertainty. The hyperparameter optimisation is quite challenging, although it is likely that the noisy gradient evaluations are the primary cause of this problem.

There is a powerful conclusion to be drawn from Chapter 4: it is possible for GPs to constructively communicate in a differentially private manner that reduces to analytical matrix mechanics. The main problems encountered are associated with how to properly integrate the predictive uncertainty of the one statistician into another's model.

Two successful schemes have been introduced, although it is not exactly clear what assumptions each is making. It is possible that neither are optimally treating the predictive uncertainty: in Section 4.5 no method successfully included the weaker predictions within the stronger GP, which would possibly have been improved by a more principled treatment of the predictive uncertainty integration.

There is an element of trust in using another statistician's predictive distribution in your own analysis. It is possible for the variance to be poorly calibrated in situations of model misspecification: it is a problem from the theoretical foundations of Bayesian analysis to ensure robustness in such circumstances. In this work, the use of nonparametric models aimed to minimise this problem, but it would be worth remembering when extending this framework to other models. It is also possible that introducing the perturbations necessary for differentially private communication results in an underestimated predictive variance, which would necessitate correction in future work.

There is more work to be done considering a more general framework for differentially private co-training: exactly where the boundaries of privacy lie will vary according to the situation, and may necessitate a more resilient defence of the privacy of the covariates. A general, adaptable system for performing differentially private inference among multiple statisticians would be of considerable value to the data science community.

Appendix A

Algorithm and Results Tables for Chapter 2

Algorithm 1 *An algorithm for multiplying a Kronecker-structured matrix* $\mathbf{A} = \bigotimes_{l=1}^{p} \mathbf{A}^{(l)}$ *with a vector* $\mathbf{b}$*, following that found in Saatci (2011).*

Precondition: *p matrices* $[\mathbf{A}^{(1)} \dots \mathbf{A}^{(p)}]$*, length-n vector* $\mathbf{b}$

1: ***function*** `kron_mvprod`$([\mathbf{A}^{(1)} \dots \mathbf{A}^{(p)}], \mathbf{b})$
2: $\quad \mathbf{x} \leftarrow \mathbf{b}$
3: $\quad$ ***for*** $l \leftarrow 1$ *to* p ***do***
4: $\quad\quad n_l \leftarrow \texttt{size}(\mathbf{A}^{(l)})$
5: $\quad\quad \mathbf{X} \leftarrow \texttt{reshape}(\mathbf{x}, n_l, n/n_l)$
6: $\quad\quad \mathbf{Z} \leftarrow \mathbf{A}^{(l)}\mathbf{Z}$
7: $\quad\quad \mathbf{Z} \leftarrow \mathbf{Z}^T$
8: $\quad\quad \mathbf{x} \leftarrow \texttt{vec}(\mathbf{Z})$
9: $\quad$ ***end for***
10: $\quad$ ***return*** $\mathbf{x}$
11: ***end function***

end

q_0	m_0	Data: Jutland	CCPP	Household
1	3	$0.933 \pm 9.91\text{e}-4$	0.399 ± 0.123	0.66 ± 0.227
	5	$0.879 \pm 9.28\text{e}-3$	0.368 ± 0.133	0.529 ± 0.297
	10	0.832 ± 0.0326	0.44 ± 0.488	0.505 ± 0.263
	20	0.816 ± 0.0432	1.04 ± 1.09	2.93 ± 3.58
2	3	$0.928 \pm 1.69\text{e}-3$	0.281 ± 0.0211	0.165 ± 0.181
	5	$0.853 \pm 7.0\text{e}-3$	0.298 ± 0.106	0.24 ± 0.233
	10	0.748 ± 0.0409	0.721 ± 0.672	1.25 ± 1.84
	20	0.797 ± 0.0832	2.69 ± 3.38	29.7 ± 46.1
3	3	$0.925 \pm 4.42\text{e}-3$	0.314 ± 0.147	0.34 ± 0.295
	5	0.842 ± 0.0134	0.355 ± 0.246	0.507 ± 0.387
	10	0.742 ± 0.0663	0.919 ± 0.995	7.3 ± 30.9
	20	0.78 ± 0.0657	8.2 ± 8.07	$434 \pm 1.5\text{e}3$

Table A.1: The test RMSEs of the TVGP, presented as means and standard deviations across runs, stratified by data set, q_0 and m_0.

q_0	m_0	Data: Jutland	CCPP	Household
1	3	$0.932 \pm 4.91\text{e}-4$	0.39 ± 0.0827	0.815 ± 0.267
	5	$0.875 \pm 3.78\text{e}-3$	0.348 ± 0.0606	0.54 ± 0.604
	10	0.817 ± 0.02	0.355 ± 0.0569	0.434 ± 0.511
	20	0.805 ± 0.0307	0.945 ± 0.349	1.11 ± 2.66
2	3	$0.928 \pm 1.17\text{e}-3$	$0.276 \pm 5.0\text{e}-3$	0.11 ± 0.0351
	5	$0.851 \pm 8.1\text{e}-3$	0.274 ± 0.0113	0.175 ± 0.074
	10	0.731 ± 0.025	0.58 ± 0.555	0.828 ± 0.154
	20	0.761 ± 0.131	1.65 ± 1.94	12.7 ± 29.3
3	3	$0.926 \pm 7.26\text{e}-3$	$0.274 \pm 7.39\text{e}-3$	0.152 ± 0.56
	5	0.844 ± 0.0142	0.274 ± 0.0146	0.379 ± 0.629
	10	0.719 ± 0.0274	0.68 ± 0.398	1.28 ± 1.01
	20	0.773 ± 0.0893	5.11 ± 9.93	103 ± 196

Table A.2: The test RMSEs of the TVGP, presented as medians and interquartile ranges across runs, stratified by data set, q_0 and m_0.

q_0	m_0	Data: Jutland	CCPP	Household
1	3	134 ± 8.33	199 ± 7.99	461 ± 16.9
	5	145 ± 8.71	232 ± 11.6	556 ± 23.1
	10	183 ± 16.4	310 ± 23.9	790 ± 46.5
	20	255 ± 30.2	486 ± 48.2	$1.29e3 \pm 95.8$
2	3	137 ± 6.51	228 ± 15.9	786 ± 250
	5	154 ± 9.25	274 ± 24.7	$1.04e3 \pm 402$
	10	197 ± 16.2	389 ± 47.9	$1.72e3 \pm 776$
	20	297 ± 34.2	639 ± 94.8	$3.16e3 \pm 1.58e3$
3	3	146 ± 9.32	263 ± 35.7	$6.47e3 \pm 5.99e3$
	5	167 ± 12.1	330 ± 53.0	$1.01e4 \pm 9.38e3$
	10	223 ± 22.0	489 ± 95.5	$2.19e4 \pm 2.13e4$
	20	337 ± 38.6	861 ± 215	$4.65e4 \pm 4.28e4$

Table A.3: The computational times in seconds of the TVGP optimisation, presented as means and standard deviations across runs, stratified by data set, q_0 and m_0.

q_0	m_0	Data: Jutland	CCPP	Household
1	3	130 ± 8.58	199 ± 12.6	460 ± 35.8
	5	143 ± 10.9	231 ± 17.6	557 ± 44.4
	10	186 ± 30.8	307 ± 37.4	793 ± 74.4
	20	256 ± 52.1	477 ± 71.1	$1.29e3 \pm 139$
2	3	137 ± 10.7	228 ± 24.9	716 ± 350
	5	154 ± 14.6	271 ± 38.6	947 ± 568
	10	196 ± 27.4	382 ± 71.8	$1.52e3 \pm 1.1e3$
	20	296 ± 63.5	619 ± 140	$2.72e3 \pm 2.16e3$
3	3	144 ± 14.0	254 ± 70.3	$4.51e3 \pm 7.72e3$
	5	164 ± 21.1	315 ± 97.7	$7.0e3 \pm 1.37e4$
	10	223 ± 39.9	468 ± 173	$1.48e4 \pm 2.87e4$
	20	336 ± 73.5	819 ± 349	$3.36e4 \pm 6.64e4$

Table A.4: The computational times in seconds of the TVGP optimisation, presented as medians and interquartile ranges across runs, stratified by data set, q_0 and m_0.

q_0	b	Data: Jutland	Data: CCPP	Data: Household
1	1	0.873 ± 0.058	0.793 ± 1.22	0.89 ± 0.826
	10	0.86 ± 0.0521	0.472 ± 0.258	1.85 ± 3.55
	50	0.863 ± 0.0509	0.485 ± 0.272	1.05 ± 1.52
	100	0.864 ± 0.0509	0.503 ± 0.26	0.835 ± 1.02
2	1	0.853 ± 0.0776	1.51 ± 3.41	9.21 ± 35.8
	10	0.834 ± 0.0792	0.869 ± 1.19	6.81 ± 21.9
	50	0.82 ± 0.0815	0.817 ± 1.22	7.94 ± 25.8
	100	0.818 ± 0.0827	0.791 ± 0.944	7.37 ± 18.4
3	1	0.845 ± 0.0816	4.43 ± 8.33	$345 \pm 1.51e3$
	10	0.817 ± 0.0854	1.73 ± 3.33	63.8 ± 141
	50	0.816 ± 0.0805	1.87 ± 3.52	31.0 ± 69.5
	100	0.811 ± 0.0836	1.76 ± 3.54	2.37 ± 6.74

Table A.5: The test RMSEs of the TVGP, presented as means and standard deviations across runs, stratified by data set, q_0 and b.

q_0	b	Data: Jutland	Data: CCPP	Data: Household
1	1	0.878 ± 0.112	0.372 ± 0.491	0.816 ± 0.359
	10	0.868 ± 0.096	0.36 ± 0.084	0.819 ± 0.245
	50	0.874 ± 0.095	0.36 ± 0.115	0.626 ± 0.544
	100	0.874 ± 0.0998	0.365 ± 0.226	0.818 ± 0.526
2	1	0.863 ± 0.121	0.382 ± 1.09	0.806 ± 1.47
	10	0.852 ± 0.172	0.299 ± 0.607	0.484 ± 2.44
	50	0.845 ± 0.158	0.298 ± 0.759	0.306 ± 0.874
	100	0.843 ± 0.147	0.291 ± 0.694	0.195 ± 0.919
3	1	0.858 ± 0.162	0.982 ± 2.26	1.14 ± 21.3
	10	0.846 ± 0.172	0.323 ± 0.738	0.82 ± 35.8
	50	0.824 ± 0.154	0.321 ± 0.829	0.812 ± 3.01
	100	0.819 ± 0.179	0.311 ± 0.89	0.83 ± 1.18

Table A.6: The test RMSEs of the TVGP, presented as medians and interquartile ranges across runs, stratified by data set, q_0 and b.

q_0	b	Data: Jutland	CCPP	Household
1	1	172 ± 40.2	282 ± 95.0	754 ± 301
	10	163 ± 39.4	291 ± 98.7	726 ± 292
	50	181 ± 54.2	315 ± 116	776 ± 324
	100	201 ± 58.1	340 ± 135	842 ± 371
2	1	177 ± 50.2	332 ± 127	921 ± 407
	10	184 ± 52.9	350 ± 136	$1.11e3 \pm 534$
	50	208 ± 71.6	399 ± 169	$1.85e3 \pm 1.02e3$
	100	217 ± 74.5	449 ± 205	$2.83e3 \pm 1.72e3$
3	1	194 ± 60.6	375 ± 157	$2.12e3 \pm 1.29e3$
	10	202 ± 65.0	408 ± 176	$5.32e3 \pm 3.67e3$
	50	239 ± 83.3	538 ± 261	$2.57e4 \pm 2.08e4$
	100	238 ± 88.4	621 ± 333	$5.18e4 \pm 3.7e4$

Table A.7: The computational times in seconds of the TVGP optimisation, presented as means and standard deviations across runs, stratified by data set, q_0 and b.

q_0	b	Data: Jutland	CCPP	Household
1	1	157 ± 58.1	251 ± 113	648 ± 385
	10	150 ± 45.8	259 ± 117	632 ± 349
	50	159 ± 69.3	278 ± 139	673 ± 390
	100	181 ± 77.7	298 ± 161	723 ± 444
2	1	161 ± 58.8	291 ± 150	781 ± 479
	10	167 ± 64.1	307 ± 163	936 ± 645
	50	181 ± 82.6	346 ± 202	$1.53e3 \pm 1.25e3$
	100	193 ± 88.3	385 ± 245	$2.27e3 \pm 2.07e3$
3	1	175 ± 74.8	326 ± 188	$1.69e3 \pm 1.55e3$
	10	182 ± 78.1	353 ± 211	$4.07e3 \pm 4.25e3$
	50	214 ± 110	446 ± 333	$1.87e4 \pm 2.33e4$
	100	210 ± 107	509 ± 386	$3.99e4 \pm 4.76e4$

Table A.8: The computational times in seconds of the TVGP optimisation, presented as medians and interquartile ranges across runs, stratified by data set, q_0 and b.

Data	b	m 10	50	200	500
Jutland	1	0.761 ± 0.0116	0.763 ± 0.0186	$0.992 \pm 3.43e-07$	$0.992 \pm 2.87e-08$
	10	0.719 ± 0.00906	0.683 ± 0.0149	$0.992 \pm 2.15e-08$	$0.992 \pm 1.31e-08$
	50	0.707 ± 0.0024	0.641 ± 0.00927	$0.992 \pm 3.69e-09$	$0.992 \pm 5.88e-09$
	100	0.704 ± 0.00322	0.624 ± 0.00912	$0.992 \pm 5.58e-09$	$0.992 \pm 1.4e-08$
CCPP	1	0.298 ± 0.0114	$0.289 \pm 9.21\text{e}-3$	0.368 ± 0.22	$0.283 \pm 5.63\text{e}-3$
	10	$0.28 \pm 3.95\text{e}-3$	$0.278 \pm 3.4\text{e}-3$	$0.277 \pm 3.62\text{e}-3$	$0.276 \pm 1.64\text{e}-3$
	50	$0.277 \pm 1.18\text{e}-3$	$0.274 \pm 8.69\text{e}-4$	$0.272 \pm 2.14\text{e}-3$	$0.273 \pm 7.08\text{e}-4$
	100	$0.276 \pm 7.8\text{e}-4$	$0.273 \pm 1.18\text{e}-3$	$0.268 \pm 2.28\text{e}-3$	$0.272 \pm 1.09\text{e}-3$
Household	1	$0.0619 \pm 4.64\text{e}-3$	0.163 ± 0.0233	0.663 ± 0.0896	0.685 ± 0.0525
	10	0.0536 ± 0.0147	0.117 ± 0.0252	0.263 ± 0.0986	0.653 ± 0.0873
	50	$0.0403 \pm 4.71\text{e}-3$	0.0858 ± 0.0197	0.121 ± 0.0225	0.348 ± 0.22
	100	$0.0396 \pm 6.24\text{e}-3$	$0.0542 \pm 6.63\text{e}-3$	0.102 ± 0.0208	0.16 ± 0.0404

Table A.9: The test RMSEs of the SVIGP, presented as means and standard deviations across runs, stratified by data set, m and b.

Data	b	m 10	50	200	500
Jutland	1	0.764 ± 0.0131	0.756 ± 0.00708	$0.992 \pm 4.64e-08$	$0.992 \pm 2.96e-08$
	10	0.717 ± 0.0151	0.681 ± 0.0156	$0.992 \pm 2.76e-08$	$0.992 \pm 2.95e-09$
	50	0.707 ± 0.00424	0.642 ± 0.00844	$0.992 \pm 5.14e-10$	$0.992 \pm 4.53e-11$
	100	0.703 ± 0.00307	0.621 ± 0.00675	$0.992 \pm 2.25e-12$	$0.992 \pm 1.38e-09$
CCPP	1	0.299 ± 0.0179	0.286 ± 0.01	0.293 ± 0.0176	$0.282 \pm 8.28\text{e}-3$
	10	$0.278 \pm 1.57\text{e}-3$	$0.277 \pm 2.8\text{e}-3$	$0.276 \pm 4.02\text{e}-3$	$0.276 \pm 1.41\text{e}-3$
	50	$0.277 \pm 1.23\text{e}-3$	$0.273 \pm 1.21\text{e}-3$	$0.272 \pm 2.6\text{e}-3$	$0.273 \pm 1.09\text{e}-3$
	100	$0.276 \pm 1.1\text{e}-3$	$0.273 \pm 1.46\text{e}-3$	$0.268 \pm 1.08\text{e}-3$	$0.272 \pm 1.37\text{e}-3$
Household	1	$0.0617 \pm 8.89\text{e}-3$	0.166 ± 0.0374	0.673 ± 0.094	0.684 ± 0.0607
	10	0.0457 ± 0.0139	0.115 ± 0.0336	0.228 ± 0.0609	0.669 ± 0.106
	50	$0.0384 \pm 4.78\text{e}-3$	0.0827 ± 0.0313	0.118 ± 0.0203	0.263 ± 0.323
	100	$0.0375 \pm 2.79\text{e}-3$	$0.0542 \pm 6.64\text{e}-3$	0.103 ± 0.0302	0.154 ± 0.0296

Table A.10: The test RMSEs of the SVIGP, presented as medians and interquartile ranges across runs, stratified by data set, m and b.

Data	b	m 10	50	200	500
Jutland	1	176 ± 14.9	198 ± 32.3	678 ± 11.1	$1.28e4 \pm 9.26e3$
	10	178 ± 18.5	180 ± 0.722	704 ± 19.3	$6.66e3 \pm 599$
	50	192 ± 9.61	203 ± 0.378	826 ± 33.7	$6.4e3 \pm 425$
	100	203 ± 16.9	232 ± 0.691	906 ± 5.13	$6.91e3 \pm 418$
CCPP	1	146 ± 0.629	174 ± 0.648	755 ± 196	$6.13e3 \pm 195$
	10	210 ± 26.9	180 ± 0.639	722 ± 1.94	$5.98e3 \pm 256$
	50	213 ± 28.4	204 ± 0.481	937 ± 69.2	$6.36e3 \pm 269$
	100	212 ± 8.02	235 ± 2.26	956 ± 5.62	$6.64e3 \pm 12.3$
Household	1	154 ± 10.6	178 ± 0.881	717 ± 7.12	$6.24e3 \pm 303$
	10	204 ± 17.8	182 ± 0.49	741 ± 7.32	$6.66e3 \pm 430$
	50	207 ± 17.8	207 ± 0.489	866 ± 46.4	$6.6e3 \pm 369$
	100	188 ± 25.2	238 ± 0.754	972 ± 4.35	$6.73e3 \pm 8.2$

Table A.11: The computational time in seconds of the optimisation of the SVIGP, presented as means and standard deviations across runs, stratified by data set, m and b.

Data	b	m 10	50	200	500
Jutland	1	177 ± 22.5	177 ± 51.5	677 ± 16.4	8.44e3 $\pm$ 6.75e3
	10	183 ± 29.4	180 ± 1.03	697 ± 4.01	6.46e3 $\pm$ 706
	50	192 ± 9.4	203 ± 0.718	814 ± 1.48	6.42e3 $\pm$ 815
	100	205 ± 19.0	232 ± 0.91	907 ± 6.45	6.77e3 $\pm$ 424
CCPP	1	147 ± 0.718	174 ± 0.919	689 ± 6.05	6.12e3 $\pm$ 232
	10	211 ± 25.6	180 ± 0.516	722 ± 2.63	5.86e3 $\pm$ 492
	50	204 ± 11.1	204 ± 0.291	953 ± 126	6.15e3 $\pm$ 465
	100	211 ± 11.7	234 ± 3.35	955 ± 10.1	6.64e3 $\pm$ 13.5
Household	1	150 ± 0.866	178 ± 0.786	716 ± 8.43	6.14e3 $\pm$ 394
	10	207 ± 24.2	182 ± 0.834	744 ± 10.9	6.62e3 $\pm$ 703
	50	212 ± 27.4	207 ± 0.817	847 ± 5.26	6.49e3 $\pm$ 526
	100	182 ± 51.6	238 ± 1.11	971 ± 7.57	6.73e3 $\pm$ 14.4

Table A.12: The computational time in seconds of the optimisation of the SVIGP, presented as medians and interquartile ranges across runs, stratified by data set, m and b.

Appendix B

Results Tables for Chapter 3

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	0.435 ± 0.0246	0.425 ± 0.0377	0.419 ± 0.0207
		10	0.382 ± 0.0264	$0.356 \pm 9.61e-3$	0.365 ± 0.015
		50	0.365 ± 0.0219	0.359 ± 0.0129	0.349 ± 0.0109
		100	0.364 ± 0.0133	$0.35 \pm 7.68e-3$	0.345 ± 0.0159
	200	1	0.424 ± 0.0301	0.403 ± 0.0355	0.386 ± 0.0253
		10	0.36 ± 0.0209	0.34 ± 0.0116	0.344 ± 0.0143
		50	0.354 ± 0.0146	0.341 ± 0.0119	$0.345 \pm 9.6e-3$
		100	0.355 ± 0.011	$0.34 \pm 8.33e-3$	$0.343 \pm 8.97e-3$
	455	1	0.414 ± 0.0274	0.397 ± 0.023	0.393 ± 0.0168
		10	0.357 ± 0.0245	$0.345 \pm 7.5e-3$	0.342 ± 0.0145
		50	0.345 ± 0.012	0.34 ± 0.0142	0.339 ± 0.0112
		100	0.344 ± 0.0121	$0.342 \pm 9.82e-3$	0.352 ± 0.0245
0.001, $1e-06$	50	1	0.423 ± 0.0221	0.404 ± 0.0217	0.398 ± 0.0165
		10	0.359 ± 0.0164	$0.346 \pm 6.61e-3$	$0.351 \pm 9.32e-3$
		50	0.355 ± 0.0187	0.346 ± 0.0107	$0.341 \pm 6.42e-3$
		100	0.357 ± 0.0134	$0.341 \pm 5.52e-3$	0.339 ± 0.0107
	200	1	0.383 ± 0.0264	0.368 ± 0.0162	0.355 ± 0.0203
		10	$0.349 \pm 7.13e-3$	$0.338 \pm 7.48e-3$	$0.338 \pm 7.45e-3$
		50	0.353 ± 0.0109	$0.34 \pm 6.02e-3$	$0.342 \pm 7.25e-3$
		100	$0.35 \pm 8.01e-3$	0.343 ± 0.0107	$0.343 \pm 6.9e-3$
	455	1	0.388 ± 0.022	0.358 ± 0.0162	0.354 ± 0.0225
		10	0.354 ± 0.0202	$0.334 \pm 8.55e-3$	$0.334 \pm 6.97e-3$
		50	$0.334 \pm 9.2e-3$	$0.328 \pm 5.74e-3$	$0.334 \pm 6.54e-3$
		100	$0.337 \pm 7.38e-3$	$0.327 \pm 5.55e-3$	0.338 ± 0.0141

Table B.1: The test RMSEs of the RNVGP run on the Boston data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as means across runs with standard deviations between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	0.431 ± 0.0187	0.433 ± 0.0323	0.419 ± 0.0276
		10	0.381 ± 0.0483	$0.355 \pm 9.21\text{e}-3$	0.36 ± 0.0197
		50	0.361 ± 0.0271	0.356 ± 0.0183	0.348 ± 0.0139
		100	0.367 ± 0.0249	0.35 ± 0.0152	0.343 ± 0.0256
	200	1	0.417 ± 0.0291	0.387 ± 0.0569	0.388 ± 0.0247
		10	0.353 ± 0.0192	0.338 ± 0.0129	0.34 ± 0.0187
		50	0.354 ± 0.01	0.34 ± 0.0142	0.343 ± 0.0131
		100	0.357 ± 0.0173	0.338 ± 0.012	0.345 ± 0.0112
	455	1	0.406 ± 0.0318	0.402 ± 0.0269	0.396 ± 0.0172
		10	0.351 ± 0.0127	0.342 ± 0.011	$0.337 \pm 9.7\text{e}-3$
		50	0.349 ± 0.0141	0.339 ± 0.0121	0.334 ± 0.0133
		100	0.346 ± 0.0178	0.34 ± 0.0117	0.347 ± 0.0113
0.001, $1e-06$	50	1	0.419 ± 0.024	0.405 ± 0.0366	$0.398 \pm 8.73\text{e}-3$
		10	0.356 ± 0.0198	$0.345 \pm 9.74\text{e}-3$	0.349 ± 0.0168
		50	0.351 ± 0.0221	0.347 ± 0.0176	$0.341 \pm 6.45\text{e}-3$
		100	0.353 ± 0.015	$0.341 \pm 7.38\text{e}-3$	0.334 ± 0.0116
	200	1	0.386 ± 0.0293	0.364 ± 0.0194	0.358 ± 0.0228
		10	0.35 ± 0.0112	$0.339 \pm 8.7\text{e}-3$	$0.341 \pm 9.7\text{e}-3$
		50	$0.353 \pm 7.26\text{e}-3$	$0.34 \pm 7.67\text{e}-3$	0.345 ± 0.0116
		100	0.349 ± 0.0136	0.342 ± 0.017	0.342 ± 0.0108
	455	1	0.385 ± 0.0217	0.357 ± 0.0238	0.35 ± 0.0141
		10	$0.347 \pm 8.39\text{e}-3$	0.337 ± 0.0135	$0.333 \pm 5.66\text{e}-3$
		50	$0.335 \pm 8.27\text{e}-3$	$0.327 \pm 9.15\text{e}-3$	$0.333 \pm 9.16\text{e}-3$
		100	$0.339 \pm 6.08\text{e}-3$	$0.326 \pm 9.35\text{e}-3$	0.334 ± 0.0171

Table B.2: The test RMSEs of the RNVGP run on the Boston data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as medians across runs with interquartile ranges between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	0.57 ± 0.0941	0.56 ± 0.0564	0.529 ± 0.064
		10	0.589 ± 0.0903	0.545 ± 0.0975	0.506 ± 0.083
		50	0.525 ± 0.105	0.516 ± 0.0683	0.47 ± 0.071
		100	0.54 ± 0.0986	0.464 ± 0.021	0.495 ± 0.0904
	200	1	0.488 ± 0.0373	0.48 ± 0.0338	0.484 ± 0.027
		10	0.475 ± 0.0213	0.473 ± 0.0267	0.458 ± 0.022
		50	0.466 ± 0.0176	0.456 ± 0.0187	0.463 ± 0.0337
		100	0.465 ± 0.0383	0.464 ± 0.0187	0.444 ± 0.0263
	500	1	0.562 ± 0.0233	0.546 ± 0.0256	0.531 ± 0.0353
		10	0.483 ± 0.0328	0.471 ± 0.0198	0.484 ± 0.0288
		50	0.484 ± 0.0226	0.476 ± 0.0247	0.467 ± 0.0339
		100	0.484 ± 0.033	0.443 ± 0.0204	0.457 ± 0.0286
0.001, $1e-06$	50	1	0.534 ± 0.0769	0.482 ± 0.0468	0.46 ± 0.0103
		10	0.546 ± 0.0937	$0.443 \pm 8.07\text{e}-3$	0.46 ± 0.0681
		50	0.474 ± 0.0613	0.461 ± 0.027	0.45 ± 0.0688
		100	0.496 ± 0.0773	0.457 ± 0.0387	0.453 ± 0.0736
	200	1	0.461 ± 0.0258	0.435 ± 0.0184	0.447 ± 0.0242
		10	0.444 ± 0.0282	0.431 ± 0.0175	$0.417 \pm 7.5\text{e}-3$
		50	0.457 ± 0.0137	0.422 ± 0.015	0.418 ± 0.0129
		100	0.433 ± 0.0257	0.418 ± 0.0152	0.413 ± 0.0134
	500	1	0.525 ± 0.0247	0.467 ± 0.0293	0.453 ± 0.0244
		10	0.454 ± 0.0198	0.415 ± 0.0201	0.424 ± 0.023
		50	0.458 ± 0.0259	0.432 ± 0.0255	0.424 ± 0.0167
		100	0.44 ± 0.0346	0.415 ± 0.0281	0.413 ± 0.0156

Table B.3: The test RMSEs of the RNVGP run on the SARCOS data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as means across runs with standard deviations between runs.

ϵ_L, ϵ	m	b	d = 1	d = 5	d = 10
0.1, 0.001	50	1	0.51 ± 0.135	0.582 ± 0.102	0.493 ± 0.0807
		10	0.6 ± 0.173	0.488 ± 0.191	0.474 ± 0.0727
		50	0.461 ± 0.158	0.493 ± 0.0456	0.449 ± 0.019
		100	0.479 ± 0.168	0.463 ± 0.0309	0.457 ± 0.052
	200	1	0.481 ± 0.0545	0.474 ± 0.0256	0.476 ± 0.034
		10	0.477 ± 0.0235	0.463 ± 0.0248	0.461 ± 0.033
		50	0.466 ± 0.0268	0.458 ± 0.0286	0.451 ± 0.0283
		100	0.454 ± 0.0434	0.467 ± 0.0241	0.445 ± 0.0365
	500	1	0.565 ± 0.0246	0.545 ± 0.0282	0.535 ± 0.0432
		10	0.471 ± 0.0619	0.471 ± 0.0252	0.48 ± 0.023
		50	0.49 ± 0.0281	0.483 ± 0.0285	0.477 ± 0.0514
		100	0.484 ± 0.0506	0.441 ± 0.0286	0.451 ± 0.0467
0.001, $1e-06$	50	1	0.493 ± 0.09	0.464 ± 0.0113	$0.456 \pm 6.33\text{e}-3$
		10	0.482 ± 0.182	$0.443 \pm 6.02\text{e}-3$	0.433 ± 0.02
		50	0.462 ± 0.0191	0.459 ± 0.0322	$0.425 \pm 8.57\text{e}-3$
		100	0.458 ± 0.0205	0.444 ± 0.0129	0.428 ± 0.0102
	200	1	0.464 ± 0.0468	0.44 ± 0.0272	0.445 ± 0.0404
		10	0.452 ± 0.0354	0.425 ± 0.0267	$0.415 \pm 7.58\text{e}-3$
		50	0.458 ± 0.0261	0.427 ± 0.0261	0.417 ± 0.0123
		100	0.432 ± 0.0188	0.417 ± 0.026	0.415 ± 0.0225
	500	1	0.521 ± 0.042	0.468 ± 0.0297	0.459 ± 0.0332
		10	0.45 ± 0.0255	0.412 ± 0.0318	0.424 ± 0.0374
		50	0.46 ± 0.0267	0.432 ± 0.0456	0.427 ± 0.0286
		100	0.445 ± 0.0641	0.423 ± 0.0465	0.414 ± 0.0248

Table B.4: The test RMSEs of the RNVGP run on the SARCOS data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as medians across runs with interquartile ranges between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	51.0 ± 0.52	98.6 ± 3.3	150 ± 7.56
		10	56.2 ± 0.441	104 ± 4.1	161 ± 5.08
		50	65.5 ± 0.911	116 ± 4.22	181 ± 5.58
		100	76.5 ± 1.15	133 ± 3.42	198 ± 5.94
	200	1	195 ± 1.97	328 ± 17.3	481 ± 19.0
		10	200 ± 4.02	373 ± 14.7	538 ± 28.3
		50	244 ± 5.13	422 ± 16.0	588 ± 30.3
		100	300 ± 2.43	479 ± 17.0	668 ± 37.9
	455	1	864 ± 20.9	$1.56e3 \pm 97.4$	$2.02e3 \pm 194$
		10	929 ± 32.0	$1.68e3 \pm 80.3$	$2.2e3 \pm 99.1$
		50	$1.03e3 \pm 23.5$	$1.91e3 \pm 145$	$1.86e3 \pm 112$
		100	$1.15e3 \pm 16.9$	$1.82e3 \pm 51.2$	$2.54e3 \pm 287$
$0.001, 1e-06$	50	1	57.4 ± 0.53	129 ± 6.15	214 ± 8.14
		10	62.4 ± 0.934	139 ± 3.94	235 ± 7.97
		50	73.3 ± 2.72	154 ± 5.72	261 ± 8.55
		100	83.9 ± 1.64	170 ± 3.77	278 ± 11.2
	200	1	217 ± 3.25	427 ± 16.8	700 ± 33.6
		10	225 ± 2.23	492 ± 12.2	785 ± 19.2
		50	269 ± 3.54	541 ± 18.6	859 ± 29.7
		100	327 ± 4.08	607 ± 13.2	941 ± 33.1
	455	1	980 ± 22.5	$2.1e3 \pm 98.7$	$2.92e3 \pm 176$
		10	$1.06e3 \pm 26.9$	$2.3e3 \pm 86.1$	$3.19e3 \pm 234$
		50	$1.16e3 \pm 30.5$	$2.5e3 \pm 93.1$	$2.97e3 \pm 145$
		100	$1.28e3 \pm 16.9$	$2.45e3 \pm 123$	$3.61e3 \pm 197$

Table B.5: The computational times in seconds of the RNVGP run on the Boston data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as means across runs with standard deviations between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	50.8 ± 0.748	99.0 ± 2.21	148 ± 10.3
		10	56.2 ± 0.601	103 ± 6.72	162 ± 7.45
		50	65.5 ± 1.46	115 ± 2.4	182 ± 6.31
		100	76.5 ± 1.83	132 ± 5.45	197 ± 9.52
	200	1	195 ± 1.61	329 ± 20.0	476 ± 20.7
		10	202 ± 4.69	371 ± 25.4	534 ± 24.0
		50	244 ± 2.86	416 ± 14.3	579 ± 25.6
		100	300 ± 2.45	470 ± 23.8	668 ± 58.6
	455	1	856 ± 22.7	$1.52e3 \pm 132$	$1.95e3 \pm 277$
		10	923 ± 32.1	$1.66e3 \pm 116$	$2.18e3 \pm 120$
		50	$1.02e3 \pm 20.2$	$1.88e3 \pm 163$	$1.89e3 \pm 117$
		100	$1.16e3 \pm 19.0$	$1.82e3 \pm 45.2$	$2.5e3 \pm 296$
0.001, $1e-06$	50	1	57.5 ± 0.617	130 ± 6.14	216 ± 12.5
		10	62.1 ± 1.37	138 ± 4.38	234 ± 15.6
		50	72.0 ± 2.67	153 ± 5.63	263 ± 12.2
		100	83.9 ± 1.14	170 ± 3.63	277 ± 19.2
	200	1	217 ± 3.18	422 ± 11.8	694 ± 39.9
		10	225 ± 3.81	490 ± 15.3	789 ± 29.4
		50	270 ± 4.31	545 ± 29.5	855 ± 28.6
		100	327 ± 4.41	606 ± 19.4	943 ± 42.1
	455	1	974 ± 20.7	$2.13e3 \pm 193$	$2.85e3 \pm 307$
		10	$1.06e3 \pm 54.4$	$2.33e3 \pm 144$	$3.2e3 \pm 384$
		50	$1.16e3 \pm 40.7$	$2.54e3 \pm 104$	$2.95e3 \pm 93.1$
		100	$1.28e3 \pm 29.8$	$2.4e3 \pm 229$	$3.6e3 \pm 278$

Table B.6: The computational times in seconds of the RNVGP run on the Boston data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as medians across runs with interquartile ranges between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	67.5 ± 0.89	121 ± 2.58	189 ± 9.11
		10	74.5 ± 0.561	131 ± 6.03	202 ± 9.76
		50	85.9 ± 0.629	144 ± 4.96	221 ± 9.16
		100	101 ± 0.512	162 ± 4.53	243 ± 12.3
	200	1	229 ± 2.5	416 ± 11.7	621 ± 31.0
		10	251 ± 5.68	459 ± 27.4	683 ± 31.0
		50	307 ± 3.97	544 ± 25.9	794 ± 42.4
		100	441 ± 5.65	680 ± 27.1	946 ± 57.5
	500	1	$1.25e3 \pm 19.4$	$2.36e3 \pm 199$	$3.34e3 \pm 237$
		10	$1.33e3 \pm 21.9$	$2.44e3 \pm 143$	$3.2e3 \pm 172$
		50	$1.48e3 \pm 28.9$	$2.6e3 \pm 97.1$	$3.04e3 \pm 349$
		100	$1.67e3 \pm 20.7$	$2.84e3 \pm 149$	$4.2e3 \pm 257$
$0.001, 1e-06$	50	1	73.0 ± 1.49	150 ± 8.37	242 ± 12.2
		10	79.8 ± 0.751	159 ± 8.82	280 ± 10.0
		50	92.0 ± 0.938	178 ± 9.24	292 ± 12.3
		100	107 ± 0.984	196 ± 5.81	318 ± 8.87
	200	1	248 ± 3.32	511 ± 19.2	828 ± 31.8
		10	273 ± 4.43	572 ± 22.2	884 ± 24.8
		50	328 ± 4.61	643 ± 21.2	984 ± 32.5
		100	462 ± 4.8	791 ± 26.1	$1.15e3 \pm 30.0$
	500	1	$1.37e3 \pm 21.7$	$2.87e3 \pm 208$	$4.22e3 \pm 468$
		10	$1.45e3 \pm 24.8$	$2.96e3 \pm 127$	$4.23e3 \pm 302$
		50	$1.61e3 \pm 22.6$	$3.23e3 \pm 181$	$4.22e3 \pm 269$
		100	$1.79e3 \pm 29.7$	$3.38e3 \pm 161$	$5.19e3 \pm 278$

Table B.7: The computational times in seconds of the RNVGP run on the SARCOS data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as means across runs with standard deviations between runs.

ϵ_L, ϵ	m	b	d 1	5	10
0.1, 0.001	50	1	67.4 ± 1.4	121 ± 4.06	186 ± 10.9
		10	74.5 ± 0.871	130 ± 9.85	202 ± 12.2
		50	85.9 ± 0.991	144 ± 9.35	223 ± 15.4
		100	101 ± 0.349	160 ± 9.0	242 ± 16.4
	200	1	228 ± 3.4	412 ± 6.08	612 ± 23.9
		10	250 ± 3.45	451 ± 25.2	672 ± 14.2
		50	307 ± 3.57	553 ± 36.1	799 ± 76.9
		100	440 ± 5.51	678 ± 35.7	954 ± 69.4
	500	1	$1.25e3 \pm 19.3$	$2.32e3 \pm 172$	$3.35e3 \pm 390$
		10	$1.32e3 \pm 37.2$	$2.38e3 \pm 209$	$3.22e3 \pm 163$
		50	$1.47e3 \pm 38.0$	$2.57e3 \pm 90.9$	$2.93e3 \pm 421$
		100	$1.67e3 \pm 15.6$	$2.85e3 \pm 176$	$4.22e3 \pm 315$
0.001, $1e-06$	50	1	72.6 ± 1.95	148 ± 9.58	241 ± 8.34
		10	79.8 ± 0.507	159 ± 12.8	278 ± 15.3
		50	91.9 ± 1.01	178 ± 15.5	291 ± 12.5
		100	106 ± 1.86	194 ± 10.3	323 ± 12.7
	200	1	248 ± 2.04	507 ± 18.2	825 ± 42.8
		10	273 ± 4.04	566 ± 34.7	886 ± 25.1
		50	328 ± 7.36	643 ± 30.2	976 ± 39.0
		100	462 ± 5.81	788 ± 41.1	$1.15e3 \pm 31.3$
	500	1	$1.37e3 \pm 27.5$	$2.87e3 \pm 244$	$4.07e3 \pm 386$
		10	$1.44e3 \pm 23.4$	$2.92e3 \pm 151$	$4.15e3 \pm 277$
		50	$1.61e3 \pm 34.5$	$3.25e3 \pm 237$	$4.09e3 \pm 448$
		100	$1.79e3 \pm 45.8$	$3.35e3 \pm 192$	$5.15e3 \pm 512$

Table B.8: The computational times in seconds of the RNVGP run on the SARCOS data set, stratified by d, b, m, ϵ_L and ϵ. The results are presented as medians across runs with interquartile ranges between runs.

Appendix C

Results Tables for Chapter 4

Cond Ind	Masking	$r_{\mathbf{Xy}}$	n_{obs} 25	50	200
No	No	0.5	0.925 ± 0.0757	0.883 ± 0.0483	0.867 ± 0.0367
		0.75	0.707 ± 0.0475	0.682 ± 0.0414	0.663 ± 0.0286
		0.9	0.479 ± 0.0403	0.456 ± 0.0337	0.438 ± 0.0219
		0.99	0.155 ± 0.0112	0.146 ± 0.00949	0.142 ± 0.00703
	Yes	0.5	1.00 ± 0.126	0.967 ± 0.0977	0.915 ± 0.0514
		0.75	0.866 ± 0.135	0.821 ± 0.110	0.731 ± 0.0615
		0.9	0.668 ± 0.180	0.649 ± 0.155	0.568 ± 0.147
		0.99	0.440 ± 0.276	0.333 ± 0.185	0.317 ± 0.240
Yes	No	0.5	0.935 ± 0.0528	0.883 ± 0.0436	0.876 ± 0.0329
		0.75	0.714 ± 0.0635	0.674 ± 0.0264	0.669 ± 0.0243
		0.9	0.468 ± 0.0473	0.442 ± 0.0207	0.442 ± 0.0164
		0.99	0.157 ± 0.0249	0.144 ± 0.00874	0.142 ± 0.00573
	Yes	0.5	1.01 ± 0.0943	0.975 ± 0.0710	0.937 ± 0.0585
		0.75	0.874 ± 0.132	0.849 ± 0.121	0.769 ± 0.100
		0.9	0.695 ± 0.158	0.635 ± 0.146	0.598 ± 0.146
		0.99	0.420 ± 0.196	0.406 ± 0.223	0.240 ± 0.114

Table C.1: The test RMSEs of the GPs with hyperparameters trained on simulated data, before co-training. Results are stratified by n_{obs}, $r_{\mathbf{Xy}}$, masking status and conditional independence of covariates.

Cond Ind	Masking	$r_{\mathbf{Xy}}$	n_{obs} 25	50	200
No	No	0.5	0.925 ± 0.0757	0.883 ± 0.0483	0.867 ± 0.0367
		0.75	0.707 ± 0.0475	0.682 ± 0.0414	0.663 ± 0.0286
		0.9	0.479 ± 0.0403	0.456 ± 0.0337	0.437 ± 0.0197
		0.99	0.155 ± 0.0112	0.146 ± 0.00949	0.142 ± 0.00695
	Yes	0.5	0.996 ± 0.126	0.967 ± 0.0979	0.916 ± 0.0516
		0.75	0.866 ± 0.135	0.822 ± 0.110	0.738 ± 0.0693
		0.9	0.670 ± 0.180	0.649 ± 0.156	0.567 ± 0.147
		0.99	0.413 ± 0.270	0.322 ± 0.183	0.316 ± 0.240
Yes	No	0.5	0.935 ± 0.0527	0.882 ± 0.0431	0.876 ± 0.0329
		0.75	0.714 ± 0.0635	0.680 ± 0.0366	0.669 ± 0.0243
		0.9	0.470 ± 0.0466	0.447 ± 0.0322	0.443 ± 0.0166
		0.99	0.157 ± 0.0226	0.145 ± 0.00734	0.142 ± 0.00565
	Yes	0.5	1.02 ± 0.115	0.979 ± 0.0744	0.937 ± 0.0586
		0.75	0.873 ± 0.133	0.850 ± 0.121	0.768 ± 0.0993
		0.9	0.695 ± 0.158	0.635 ± 0.146	0.598 ± 0.146
		0.99	0.433 ± 0.208	0.366 ± 0.171	0.269 ± 0.116

Table C.2: The test RMSEs of the GPs with hyperparameters trained on simulated data, before differentially private co-training. Results are stratified by n_{obs}, $r_{\mathbf{Xy}}$, masking status and conditional independence of covariates.

Bibliography

Louis JM Aslett, Pedro M Esperança, and Chris C Holmes. A Review of Homomorphic Encryption and Software Tools for Encrypted Statistical Machine Learning. Technical report, University of Oxford, 2015.

José M Bernardo and Adrian FM Smith. Bayesian Theory, 2001.

Marta Blangiardo, Michela Cameletti, Gianluca Baio, and Håvard Rue. Spatial and Spatio-temporal Models with R-INLA. *Spatial and spatio-temporal epidemiology*, 7:39–55, 2013.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Leo Breiman. Bagging Predictors. *Machine learning*, 24(2):123–140, 1996.

Merlise Clyde and Edwin S Iversen. Bayesian Model Averaging in the M-open Framework. *Bayesian Theory and Applications*, page 483, 2013.

David R Cox. Regression Models and Life-tables. In *Breakthroughs in statistics*, pages 527–541. Springer, 1992.

Zhenwen Dai, Andreas Damianou, James Hensman, and Neil Lawrence. Gaussian Process Models with Parallelization and GPU Acceleration. *arXiv preprint arXiv:1410.4984*, 2014.

John C Duchi, Michael I Jordan, and Martin J Wainwright. Local Privacy, Data Processing Inequalities, and Statistical Minimax Rates. *arXiv preprint arXiv:1302.3203*, 2013.

Filippone and Engler. Enabling Scalable Stochastic Gradient-based Inference for Gaussian Processes by Employing the Unbiased LInear System SolvEr (ULISSE). *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Gal, van der Wilk, and Rasmussen. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. *arXiv preprint arXiv:1402.1389*, 2014.

Yarin Gal and Mark van der Wilk. Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models–a Gentle Tutorial. *arXiv preprint arXiv:1402.1412*, 2014.

Zoubin Ghahramani and Hyun-Chul Kim. Bayesian Classifier Combination. 2003.

Peter Grünwald. The Safe Bayesian. In *International Conference on Algorithmic Learning Theory*, pages 169–183. Springer, 2012.

Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Differential Privacy for Functions and Functional Data. *Journal of Machine Learning Research*, 14(Feb):703–727, 2013.

Trevor J Hastie and Robert J Tibshirani. *Generalized Additive Models*, volume 43. CRC press, 1990.

James Hensman, Nicolò Fusi, and Neil Lawrence. Gaussian Processes for Big Data. 2013.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. Robust Gaussian Process Regression with a Student-t Likelihood. *The Journal of Machine Learning Research*, 12:3227–3257, 2011.

Hyunjik Kim, Xiaoyu Lu, Seth Flaxman, and Yee Whye Teh. Tucker Gaussian Process for Regression and Collaborative Filtering. *arXiv preprint arXiv:1605.07025*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

Peter McCullagh. Generalized linear models. *European Journal of Operational Research*, 16 (3):285–292, 1984.

Arman Melkumyan and Fabio Ramos. A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets. In *IJCAI*, volume 9, pages 1936–1942, 2009.

Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David Dunson. Scalable and Robust Bayesian Inference via the Median Posterior. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1656–1664, 2014.

Kristine Monteith, James L Carroll, Kevin Seppi, and Tony Martinez. Turning Bayesian model averaging into Bayesian model combination. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2657–2663. IEEE, 2011.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078, 2008.

Thomas Nickson, Tom Gunter, Chris Lloyd, Michael A Osborne, and Stephen Roberts. Blitzkriging: Kronecker-structured Stochastic Gaussian Processes. *arXiv preprint arXiv:1510.07965*, 2015.

Kamal Nigam and Rayid Ghani. Analyzing the Effectiveness and Applicability of Co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

Manfred Opper and Cédric Archambeau. The Variational Gaussian Approximation Revisited. *Neural computation*, 21(3):786–792, 2009.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *NIPS*, volume 3, page 5, 2007.

Rasmussen and Williams. *Gaussian Processes for Machine Learning,*.

Christian P Robert. *Monte Carlo Methods*. Wiley Online Library, 2004.

Sebastian Ruder. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Yunus Saatci. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

Robert E Schapire. The Boosting Approach to Machine Learning: An Overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast Forward Selection to Speed up Sparse Gaussian Process Regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

Edwin Simpson, Stephen J Roberts, Arfon Smith, and Chris Lintott. Bayesian Combination of Multiple, Imperfect Classifiers. Technical report, University of Oxford, 2011.

Michael Thomas Smith, Max Zwiessele, and Neil D Lawrence. Differentially Private Gaussian Processes. *arXiv preprint arXiv:1606.00720*, 2016.

Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

Michalis K Titsias and Miguel Lázaro-Gredilla. Variational Heteroscedastic Gaussian Process Regression. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 841–848, 2011.

Volker Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

Ledyard R Tucker. Some Mathematical Notes on Three-mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.

Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

Andrew Gordon Wilson and Hannes Nickisch. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

David H Wolpert. Stacked Generalization. *Neural networks*, 5(2):241–259, 1992.

Chang Xu, Dacheng Tao, and Chao Xu. A Survey on Multi-view Learning. *arXiv preprint arXiv:1304.5634*, 2013.

Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.

Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao. Bayesian Co-training. *The Journal of Machine Learning Research*, 12:2649–2680, 2011.